Towards accurate and efficient live cell imaging data analysis

D i s s e r t a t i o n

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

im Fach
Biophysik

eingereicht an
der Lebenswissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von

M.Eng. Hongqing Han

Präsidentin der Humboldt-Universität zu Berlin
Prof. Dr.-Ing. Dr. Sabine Kunst

Dekan der Lebenswissenschaftlichen Fakultät der Humboldt-Universität zu Berlin
Prof. Dr. rer. nat. Bernhard Grimm

Gutachter/innen
1. Prof. Dr. Dr. h.c. Edda Klipp
2. Dr. Zhike Zi
3. Dr. Edda Schulz

Tag der mündlichen Prüfung: 05.06.2020

Abstract

Microscopy offers the chance to observe the structural components and morphological characteristics of single cells, the spatial distributions of chemical species inside them, as well as their locations within native environments. Although rich in information, the recorded microscopy images are hard to interpret. Converting the matrix-like images into biological insights requires sophisticated computational pipelines consisting of multiple modules. Therefore in imaging-related studies, the reliability and efficiency of image analyses are as important as the rigor and timeliness of experiments. Image analysis can be carried out either manually or automatically. The manual approach gives scientists full control over the details during the execution of the task, and can achieve satisfactorily high accuracy, but it has poor reproducibility and scalability. The automatic approach could be effortless, fast and reproducible, but the reliability is questionable.

As an example, live cell imaging based on time-lapse microscopy has been used to study dynamic cellular behaviors, such as cell differentiation, cell cycle, cell signaling and transcription. Extracting cell lineage trees out of a time-lapse video requires cell segmentation and cell tracking. Here, the dilemma of choice between manual and automatic analyses also applies. For long term live cell imaging, data analysis errors are particularly fatal. Even an extremely low error rate could potentially be amplified by the large number of sampled time points and render the entire video useless. This calls for a well-designed computational workflow, manual or automatic, that achieves a manual-level accuracy and has an acceptable scalability.

In this work, we adopt a straightforward but practical design that combines the merits of manual and automatic approaches. We present a live cell imaging data analysis tool 'eDetect', which uses post-editing to complement automatic segmentation and tracking. What makes this work special is that eDetect employs multiple interactive data visualization modules to guide and assist users, making the error detection and correction procedure rational and efficient. Specifically, two

scatter plots and a heat map are used to interactively visualize single cells' visual features. The scatter plots position similar results in close vicinity, making it easy to spot and correct a large group of similar errors with a few mouse clicks, minimizing repetitive human interventions. The heat map is aimed at exposing all overlooked errors and helping users progressively approach perfect accuracy in cell lineage reconstruction. Quantitative evaluation proves that eDetect is able to largely improve accuracy within an acceptable time frame, and its performance surpasses the winners of most tasks in the 'Cell Tracking Challenge', as measured by biologically relevant metrics.

Zusammenfassung

Mikroskopie ermöglicht es, Morphologie und Strukturkomponenten einzelner Zellen direkt zu beobachten. Sie erlaubt es sogar, die Verteilung biochemischer Moleküle innerhalb dieser Zellen zu visualisieren und deren Lokalisation in ihrer natürlichen Umgebung direkt zu beobachten. Obwohl solche Mikroskopiebilder also sehr viele Informationen enthalten, sind sie schwer zu interpretieren. Um aus diesen digitalen Bildern, die aus Matrizen voller Zahlenreihen bestehen, biologische Erkenntnisse zu extrahieren werden ausgefeilte Rechenpipelines mit mehreren Modulen benötigt. Für mikroskopie-basierte Studien sind deshalb die Zuverlässigkeit und die Effizienz der Bildanalyse ebenso wichtig wie das experimentelle Design. Die Bildanalyse kann entweder manuell oder automatisch durchgeführt werden. Mit dem manuellen Ansatz behalten die Wissenschaftler die Kontrolle über alle Details der Analyse. Außerdem ist eine hohe Genauigkeit garantiert. Auf der anderen Seite ist die manuelle Analyse schlecht reproduzierbar und skalierbar. Der automatische Ansatz ist viel weniger arbeitsintensiv, schneller und reproduzierbarer, aber seine Zuverlässigkeit ist fraglich.

Um zum Beispiel dynamische zelluläre Prozesse wie Zelldifferenzierung, Zellzyklus, Signaltransduktion oder Transkription zu analysieren wird Live-cell-imaging mittels Zeitraffermikroskopie verwendet. Um nun aber Zellabstammungsbäume aus einem Zeitraffervideo zu extrahieren, müssen die Zellen segmentiert und verfolgt werden können. Wieder ist der Wissenschaftler mit demselben Dilemma der Wahl zwischen manuellen und automatischen Analysen konfrontiert. Besonders hier, wo lebende Zellen über einen langen Zeitraum betrachtet werden, sind Fehler in der Analyse fatal: Selbst eine extrem niedrige Fehlerrate kann sich amplifizieren, wenn viele Zeitpunkte aufgenommen werden, und damit den gesamten Datensatz unbrauchbar machen. Die Analyse solcher Fluoreszenz-Zeitraffer-Mikroskopie Daten erfordert also eine gut konzipierte Auswertung, die die Genauigkeit einer manuellen Analyse erreicht aber gleichzeitig gut skalierbar ist.

In dieser Arbeit verwenden wir einen einfachen aber praktischen Ansatz, der die

Vorzüge der manuellen und automatischen Ansätze kombiniert. Das von uns entwickelte Live-cell-Imaging Datenanalysetool ‘eDetect’ ergänzt die automatische Zellsegmentierung und -verfolgung durch Nachbearbeitung. Das Besondere an dieser Arbeit ist, dass sie mehrere interaktive Datenvisualisierungsmodule verwendet, um den Benutzer zu führen und zu unterstützen. Dies erlaubt den gesamten manuellen Eingriffsprozess zu rational und effizient zu gestalten. Insbesondere werden zwei Streudiagramme und eine Heatmap verwendet, um die Merkmale einzelner Zellen interaktiv zu visualisieren. Die Streudiagramme positionieren ähnliche Objekte in unmittelbarer Nähe. So kann eine große Gruppe ähnlicher Fehler mit wenigen Mausklicks erkannt und korrigiert werden, und damit die manuellen Eingriffe auf ein Minimum reduziert werden. Die Heatmap ist darauf ausgerichtet, alle übersehenen Fehler aufzudecken und den Benutzern dabei zu helfen, bei der Zellabstammungsrekonstruktion schrittweise die perfekte Genauigkeit zu erreichen. Die quantitative Auswertung zeigt, dass eDetect die Genauigkeit der Nachverfolgung innerhalb eines akzeptablen Zeitfensters erheblich verbessern kann. Beurteilt nach biologisch relevanten Metriken, übertrifft die Leistung von eDetect die derer Tools, die den Wettbewerb ‘Cell Tracking Challenge’ gewonnen haben.

Contents

Chapter 1

Overview

1.1 Motivation

In molecular and cell biology, it is often interesting to know the location, size, shape and structure of microscopic organisms, tissues, cells and subcellular compartments, the levels and spatial distributions of chemical species within those structures, as well as how they change over time. Using microscope systems, this information is encoded in optical signal, which is transmitted and recorded in the form of digital images.

Modern automated microscopes are able to generate voluminous imaging data, with possibly multiple channels, planes, positions and time points. This makes it prohibitive to manually extract the desired information from images. In addition, the images could also be subject to various noises, biases and artefacts, resulting from batch effect, imperfect imaging conditions and unexpected events. This largely challenges the accuracy of automatic image analysis.

Accurate interpretation of complex image datasets and retrieval of information are fundamental to drawing correct biological conclusions. At the same time, automation of data analysis is essential for the efficiency and reproducibility of biological studies. This dilemma, posed by the two challenges, calls for a solution that combines both manual and automatic approaches under optimized and customized design principles.

In this dissertation, I will discuss my efforts, attempts and explorations in designing computational workflows to find a balance between reliability and efficiency.

1.2 Outline

The content of this dissertation is organized as follows. In chapter 2, I will briefly introduce image processing and computer vision and their connections to biological image analysis. This will help the reader understand the rest of this thesis better. In chapter 3, I will present a live cell imaging data analysis tool 'eDetect' [1]. In chapter 4, I will summarize the results and discuss future directions.

Chapter 2

Background: image processing, computer vision and biological image analysis

2.1 Introduction

In 1665, Robert Hooke published a famous collection of drawings of plants, insects, and other samples, observed with microscopes - *Micrographia* [2,3], in which he named what are now known as 'cells' after what they resembled - the units of a honeycomb. This book revealed a world of miniature organic bodies to the public. Since then, microscopes have been widely employed for observing minute life phenomena, and have become an essential and powerful tool in life sciences. Nowadays, numerous molecular profiling methods have been invented, making biology more quantitative. However, microscopy is still indispensable for examining phenotypes, acquiring the spatial organization of structures and species, as well as studying live cells.

The micrographs that modern microscopes generate are digital images. Despite their various formats and dimensions due to specific instruments and imaging setups, they are essentially not different from the pictures and videos we take in everyday life, or the ones taken for industrial and security purposes. Moreover, the typical tasks in microscopy image analysis are comparable to the ones in computer vision. Therefore, discussing computer vision will help the reader better understand methods and tools for microscopy image analysis.

In this chapter, I will first prepare the reader with essential preliminary knowledge by briefly introducing the concept of digital images (section 2.2), explaining a few common techniques in digital image processing (section 2.3), and describing the usage of light microscopy in biology (section 2.4). Then I will discuss in details the typical tasks in computer vision, and their similarities and differences with their counterparts in microscopy image analysis (section 2.6). In particular, segmentation (subsection 2.6.3) and tracking (subsection 2.6.4) are the cores of live cell imaging data analysis, which is the focus of the next chapter (chapter 3). In section 2.7 I will review a few live cell imaging applications in studying cell signaling, cell cycle and transcriptional bursting.

2.2 Digital images

2.2.1 Analog signals and digital signals

Signals are descriptions of physical phenomenon - force, temperature, sound, light and electricity - in time. In other words, signals are functions that take the following form: $s = f(t), t \in T, s \in S$, meaning the value of the signal s at time t follows the mapping f, the domain and codomain of f being T and S.

Depending on the sampling and values, signals are usually catalogued into two classes: **analog signals** and **digital signals**. Analog signals (Figure 2.1 red curve) are usually continuously defined (T is uncountable) and valued (S is uncountable), while digital signals (Figure 2.1 blue squares) are discretely defined (T is countable) and valued (S is countable). In the examples shown in Figure 2.1, for the analog signal $T = [0, 100]$ and $S = [20, 60]$, while for the discrete signal $T = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ and $S = \{20, 30, 40, 50, 60\}$.

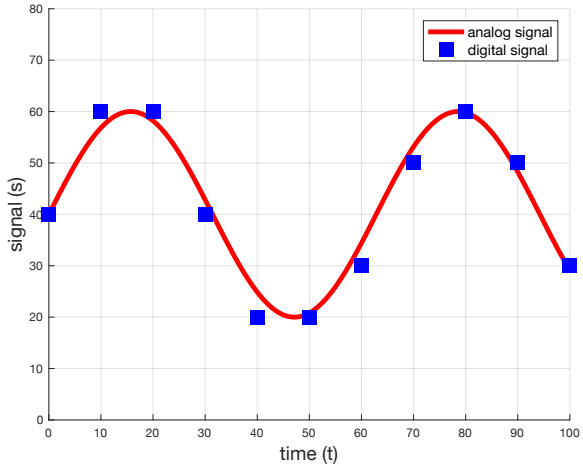


Figure 2.1: Analog signal and digital signal. t: time, s: signal value. Red curve: analog signal. Blue squares: digital signal.

Because of this difference, analog signals are not suitable for explicit representations in the form of numerical values, either written on a piece of paper or recorded in computers. For analog signals, because T is uncountable, there will be infinitely many sampled values; and because S is uncountable, most of the possible values (irrational numbers like π, e and $\sqrt{2}$) cannot be represented as fractions, and their decimal representations never terminate. Therefore, neither the representation of each value nor the number of values of analog signals is suitable for any sort of numerical recording. Instead, analog signals are usually implicitly coded

using another type of time-dependent quantity (e.g. electricity), which is analogous to the original one, and stored in media like tape or phonograph record. Explicit representation of an analog signal is only possible either analytically (e.g. $s = f(t) = \sin(2t - \pi) + 4\cos(t)$), or if it is approximated with a discrete signal of high sampling rate and numerical precision level.

Wide usage of modern computers, which use binary values ('0's and '1's) to encode and store any sort of information, makes digital signals particularly convenient to preserve and exchange.

2.2.2 Digital images are arrays of numbers

Images are recordings or simulations of light. They are a special type of signal that varies in space instead of time ($s = f(x, y)$ in which x and y represent spatial coordination). This connection between images and signals makes many signal processing concepts and methods applicable to image processing.

Like analog and digital signals, **analog images** are represented analogously and stored in media, while **digital images** are explicitly coded and are able to be stored in computers. Therefore **digital images** are more convenient than **analog images** for storage and sharing. Digital images include two classes: **vector images** and **bitmaps**. A vector image contains a set of objects such as line segments, curves, ovals and polygons, each described by a range of geometrical properties including location, shape, size and color. Vector images are comparable to the analytical representation of analog signals in the previous subsection. In this thesis, 'digital image' refers to bitmap.

A bitmap is a matrix whose two dimensions are 'height' and 'width' (Figure 2.2 A). The elements of the matrix are called **pixels** (Figure 2.2 A and B). The value of each pixel indicates the intensity or brightness of the light that comes from the location or direction that this pixel represents. The brighter a pixel appears, the higher its value is (Figure 2.2 A and C - E). Because the intensity or brightness of light cannot be negative, the values of pixels are always nonnegative (Figure 2.2 B - E). Usually, these pixel values are nonnegative integers within a limited range, and are represented as unsigned binary integers in computer systems.

In computers, integers are encoded as fixed-length binary numbers with each bit either '0' or '1'. For example, the 8-bit binary form of '19' is '00010011' ($19_{10} = 00010011_2$), meaning $19 = 0 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$. An 8-bit integer has 256 (2^8) possible combinations, and is able to represent numbers from 0_{10} (00000000_2) to 255_{10} (11111111_2). If an image's pixel

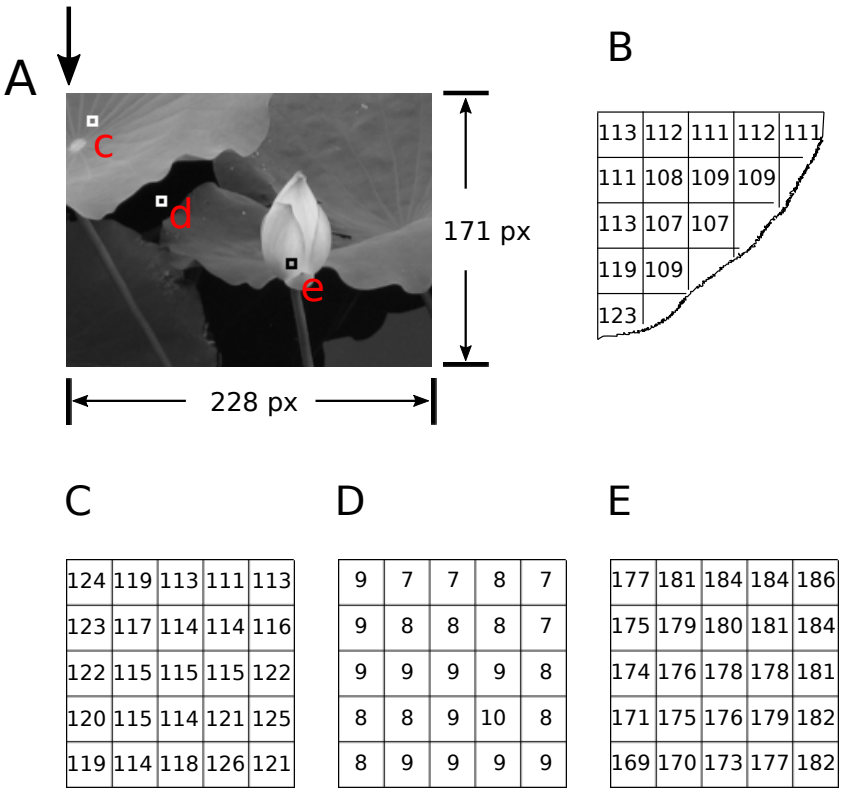


Figure 2.2: A single channel digital image is a matrix of light intensities. (A) A picture of a lotus bud. The width and height of this picture are shown alongside the picture. The values of top-left most pixels (marked by a black arrow) are shown in B. The pixel values of the square regions labeled by 'c', 'd' and 'e' are shown in C, D, and E, respectively. (B) The picture in A presented as a matrix. Only the top-left most pixels are shown for simplicity. (C)-(E) The pixel values of 5 x 5 square regions as marked in A.

values are 8-bit unsigned integers, the image has a bit-depth of 8, and its pixel values range from 0 to 255. If an image's pixel values are 16-bit unsigned integers, the image has a bit-depth of 16, and its pixel values range from 0 to 65535 (Table 2.1).

This system is sufficient when we only want to code for nonnegative integers. Otherwise, the first bit is used to indicate whether the sign of the number is positive ('0') or negative ('1'). This is why there are two classes of data types: unsigned integers and signed integers. Because pixel values are nonnegative, signed integers are not commonly used for digital images.

Table 2.1: Data types of digital images

Data type	Range	MATLAB	Python numpy	used for images
8-bit integer	-128 to 127	int8	int8	no
8-bit unsigned integer	0 to 255	uint8	uint8	yes
16-bit integer	-32768 to 32767	int16	int16	no
16-bit unsigned integer	0 to 65535	uint16	uint16	yes
boolean	0 to 1	logical	bool	yes

One integer is not sufficient to represent color of a pixel. A color needs to be described with a combination of multiple prime colors, which usually are red, green and blue. This is why **colored images** have an additional dimension - channel, whose values are usually R (red), G (green) and B (blue) (Figure 2.3 A and B). For each pixel, the value of each channel indicates the brightness of this color component in the composition of light. For example, in Figure 2.3, the second channel in the square of Figure 2.3 C has higher values than the other two channels, and the square appears green; all channels in the square of Figure 2.3 D have low values, and the square appears dark; all channels in the square of Figure 2.3 E have high values, and the square appears bright.

Those images without this additional dimension store only one value for each pixel (Figure 2.2). This single value is only able to indicate the brightness of a pixel but cannot represent the color. When a single-channeled image is to be displayed on an RGB screen, what is actually displayed is a new RGB image, whose RGB values at each pixel all equal to the single pixel value in the original image. This balanced mixture of R, G and B makes single-channeled images look colorless (gray). In fact, they are also called 'grayscale' images and their pixel values are called the grayscales.

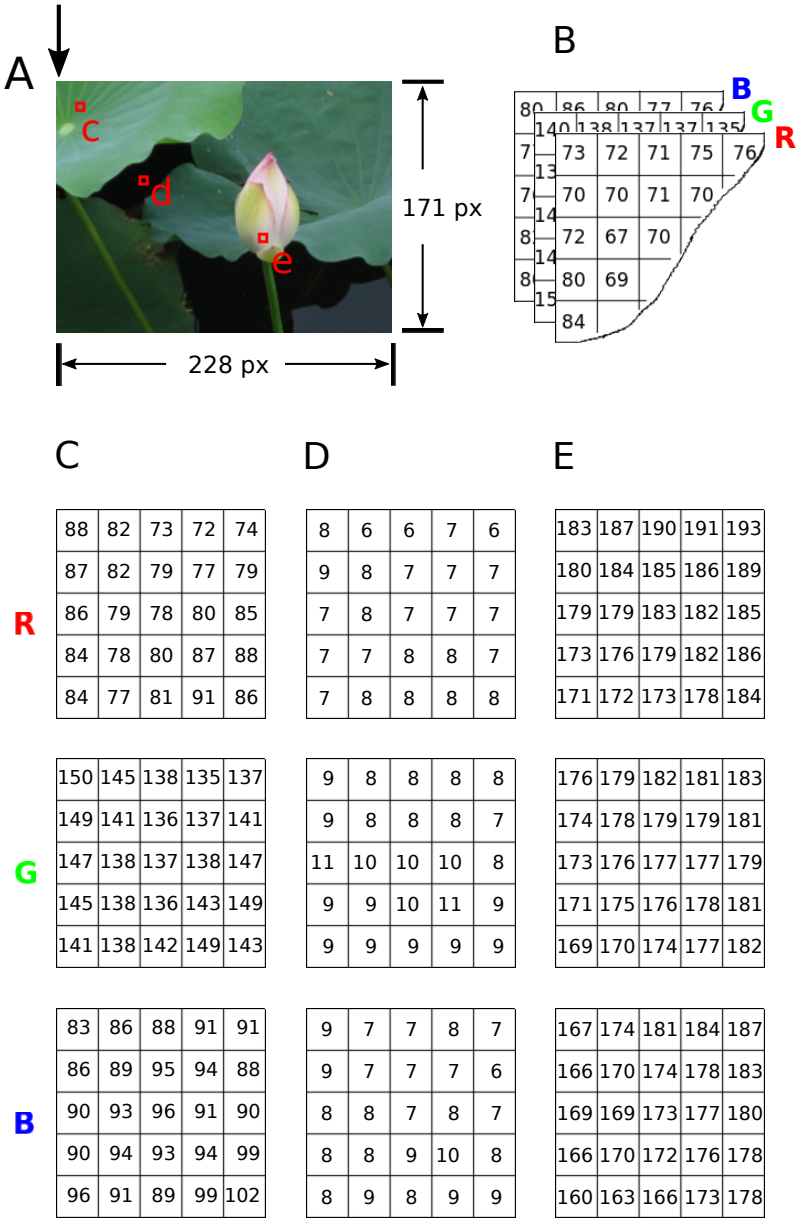


Figure 2.3: A multi-channel digital image is an array of light intensities. (A) A colored picture of a lotus bud. The width and height of this picture are shown alongside the picture. The values of top-left most pixels (marked by a black arrow) are shown in B. The pixel values of the square regions labeled by 'c', 'd' and 'e' are shown in C, D, and E, respectively. (B) The picture in A presented as a 3 dimensional array. The third dimension (apart from x and y) is channel, and it has three values: red ('R'), green ('G') and blue ('B'). Only the top-left most pixels are shown for simplicity. (C)-(E) The pixel values of 5 $\times$ 5 square regions as marked in A. From top to bottom, the 3 columns represent red, green and blue channels.

2.3 Digital image processing

Digital image processing is a computational procedure that takes digital images as input, applies computer algorithms on them, and generates new digital images as output. This definition covers a broad range of techniques such as filtering (convolution), binarization, and affine transformation. They serve various purposes including image denoising (subsection 2.3.1), image enhancement and feature extraction. Conventional computer vision is largely built on the combinatorial use of the aforementioned techniques. More recent advancements in computer vision benefit heavily from convolutional neural networks, in which the kernels (subsection 2.3.2) are not engineered by experts but instead learned from data. In addition, machine learning based methods commonly employ image processing techniques in pre-processing, post-processing or intermediate steps. For example, in the very popular segmentation method U-Net [4], elastic deformation is used in the data augmentation procedure.

In this section, I will briefly introduce a few very common digital image processing operations.

2.3.1 Median filter

Median filter is a filter commonly used to remove noise from digital signals. In the filtered signal, each element equals the median of elements within a neighborhood in the original signal. This neighborhood is also called the window of the median filter. In 1-dimensional signals, this neighborhood includes a number of elements before and after this element and itself. For example, suppose $x = (66, 3, 8, 93, 68)$, the size of neighborhood is 3, and y is the result of applying a median filter on x. To keep the size of the output signal y same as the original signal x, zeros are padded at both ends. Then $y = (3, 8, 8, 68, 68)$ because the median of $(0, 66, 3)$ is 3, the median of $(66, 3, 8)$ is 8, the median of $(3, 8, 93)$ is 8, the median of $(8, 93, 68)$ is 68, and the median of $(93, 68, 0)$ is 68.

2-dimensional median filters are used in image processing. In a filtered image, each element equals the median of pixels within a neighborhood in the original image. For example, applying a median filter of window size 3 on Figure 2.2 C yields

$$\begin{bmatrix} 124 & 119 & 113 & 111 & 113 \\ 123 & 117 & 114 & 114 & 116 \\ 122 & 115 & 115 & 115 & 122 \\ 120 & 115 & 114 & 121 & 125 \\ 119 & 114 & 118 & 126 & 121 \end{bmatrix} \xrightarrow{3\times3 \text{ median filter}} \begin{bmatrix} 0 & 114 & 113 & 113 & 0 \\ 117 & 117 & 115 & 114 & 113 \\ 115 & 115 & 115 & 115 & 115 \\ 115 & 115 & 115 & 121 & 121 \\ 0 & 114 & 114 & 118 & 0 \end{bmatrix}. \quad (2.1)$$

Zeros were padded around the boundaries so that the output matrix is of size 5

by 5. For example, the element in the middle of the output matrix is 115. This is because its neighborhood in the input matrix is,

$$\begin{bmatrix} 117 & 114 & 114 \\ 115 & 115 & 115 \\ 115 & 114 & 121 \end{bmatrix}, \tag{2.2}$$

and if we sort them in ascending order we get (114, 114, 114, 115, 115, 115, 115, 117, 121), where the median (5th) value is 115.

Figure 2.4 shows the effect of median filters of different neighborhood sizes on the same input image (Figure 2.4 A or Figure 2.2 A). The neighborhood sizes are 3×3, 9×9, and 15×15 in B, C, and D, respectively. The larger the neighborhood size is, the more blurred the output image is.

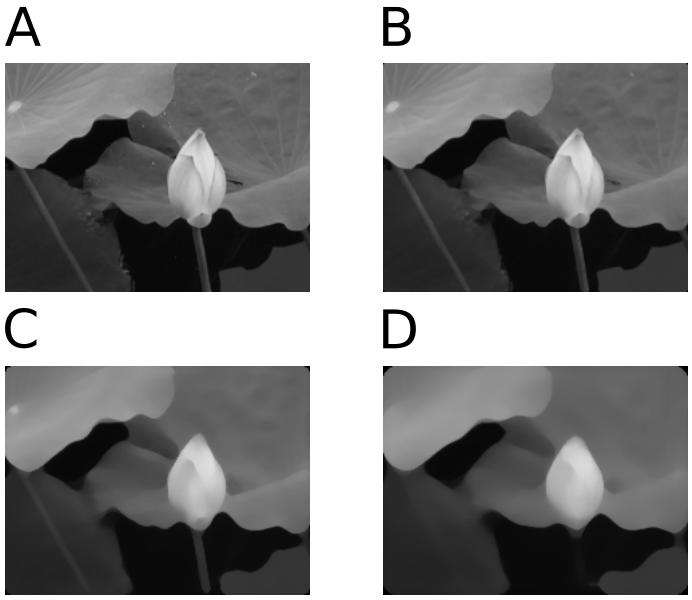


Figure 2.4: Median filtering with different neighborhood sizes. Zeros are padded around the boundary of the image so that the output images have the same size as input. (A) Original image. (B) Median filtered with 3×3 neighborhood. (C) Median filtered with 9×9 neighborhood. (D) Median filtered with 15×15 neighborhood.

2.3.2 Convolution

The convolution of two signals is a sliding inner product of one against the horizontally flipped version of the other. The convolution function $f * g$ of two continuous time signals f and g is defined as

$$\begin{aligned}(f * g)(t) &= \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau, \\ &\text{for} \quad f, g, f * g : \mathbb{R} \to \mathbb{R}.\end{aligned} \tag{2.3}$$

The convolution function $f * g$ of two discrete time (but continuous valued) signals f and g is defined as

$$\begin{aligned}(f * g)[n] &= \sum_{n=-\infty}^{\infty} f[m] g[n-m], \\ &\text{for} \quad f, g, f * g : \mathbb{Z} \to \mathbb{R}.\end{aligned} \tag{2.4}$$

The definition of convolution in digital image processing usually involves one image, and one small image called the kernel. Similar to 1-d signals, the convolution O of the input image I and the kernel $K_{(2a+1)\times(2b+1)}$ $(a, b \in \mathbb{N})$ is a sliding-window inner product of I and the horizontally and vertically flipped version of K:

$$O_{n,m} = \sum_{s=-a}^{a} \sum_{t=-b}^{b} I_{n-s,m-t} \times K_{a+1+s,b+1+t}. \tag{2.5}$$

Let's again take Figure 2.2 C as an example. Suppose

$$I = \begin{bmatrix} 124 & 119 & 113 & 111 & 113 \\ 123 & 117 & 114 & 114 & 116 \\ 122 & 115 & 115 & 115 & 122 \\ 120 & 115 & 114 & 121 & 125 \\ 119 & 114 & 118 & 126 & 121 \end{bmatrix}, K = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, O = I * K. \tag{2.6}$$

Then the element of O in the center is

$$\begin{aligned} & O_{3,3} = \\ & 117 \times 9 + 114 \times 8 + 114 \times 7 + \\ & 115 \times 6 + 115 \times 5 + 115 \times 4 + \\ & 115 \times 3 + 114 \times 2 + 121 \times 1 \\ & = 5182. \end{aligned} \tag{2.7}$$

2.3.3 Kernels

Convolution of an image with different types of kernels is able to add various types of effects onto the image. The kernels used to produce Figure 2.5 A-J are

$$K_A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.8}$$

$$K_B = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, K_C = 10K_A - 9K_B = \frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \tag{2.9}$$

$$K_D = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, K_E = 17K_A - 16K_D = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 13 & -2 \\ -1 & -2 & -1 \end{bmatrix}, \tag{2.10}$$

$$K_F = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \tag{2.11}$$

$$K_G = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, K_H = K_G^T = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \tag{2.12}$$

$$K_I = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, K_J = K_I^T = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}. \tag{2.13}$$

Kernels K_A to K_E all sum up to 1. They add effects onto the image by sharpening (K_C and K_E) or unsharpening (K_B and K_D). The sharpening kernels are derived by subtracting unsharpening kernels out of the identity kernel (K_A) with certain weight factors (10, 9, 17 and 16). Kernels K_F to K_J all sum up to 0. They are edge detectors. K_G and K_H are a pair of Prewitt operators detecting horizontal and vertical edges [5]. K_I and K_J are a pair of Sobel operators detecting horizontal and vertical edges [6].

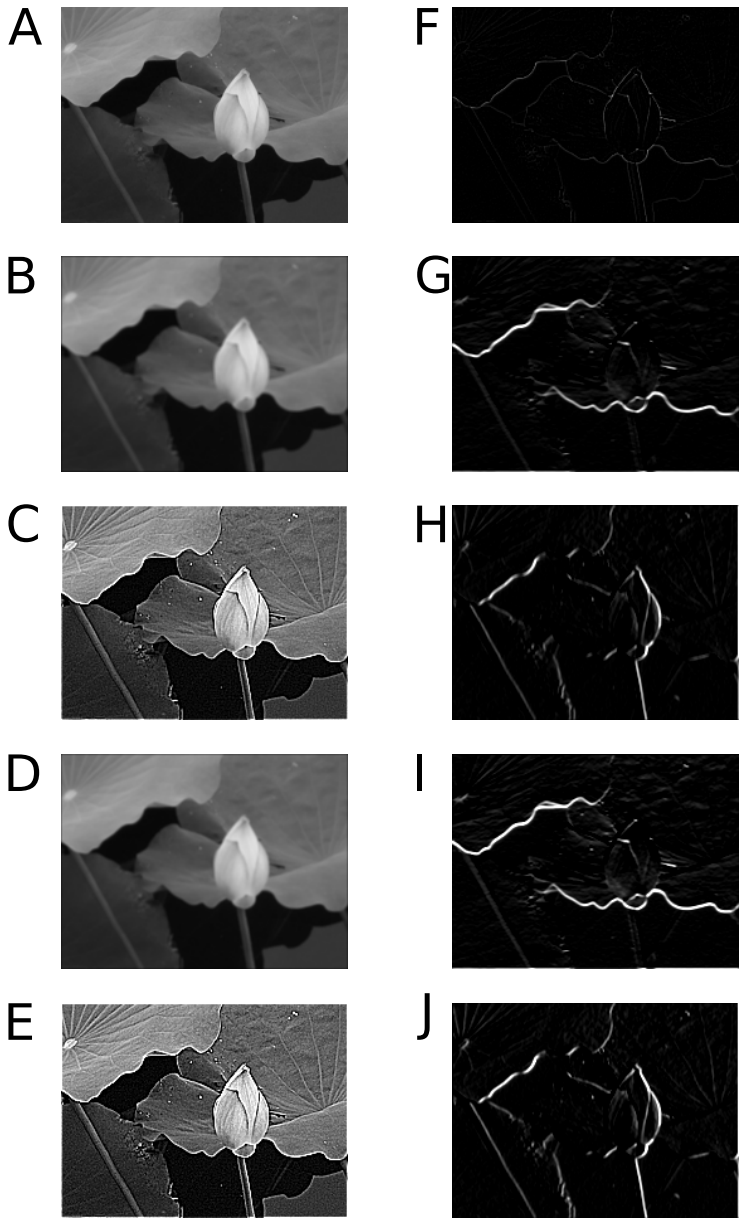


Figure 2.5: Filtering image with different convolutional kernels. (A) Identity image. (B) Mean filtered with 3×3 neighborhood. (C) Sharpened with unsharp kernel in B. (D) Approximate Gaussian filtered with 3×3 neighborhood. (E) Sharpened with unsharp kernel in D. (F) Edge detection with Laplacian kernel with alpha of 0.5. (G) Horizontal edge detection with Prewitt operator. (H) Vertical edge detection with Prewitt operator. (I) Horizontal edge detection with Sobel operator. (J) Vertical edge detection with Sobel operator.

2.3.4 Histograms

A histogram of an image is the histogram of all its pixel values. Figure 2.6 B is the histogram of the grayscale image (Figure 2.6 A).

Mapping the histogram of an image onto a new distribution is often able to enhance the contrast. For example, Figure 2.6 C is generated by applying curve adjustment on Figure 2.6 A: pixel values lower than 1st percentile of the original histogram become minimum value (0), pixel values higher than 99th percentile of the original histogram become maximum value (255), and the pixel values in between follow a linear interpolation. Figure 2.6 D is the histogram of Figure 2.6 C. Histogram equalization - transforming an image histogram into uniform distribution - is also often used for enhancing the contrast of images. Figure 2.6 E is generated from Figure 2.6 A in this way. Figure 2.6 F is the histogram of Figure 2.6 E. It approximates a uniform distribution.

2.3.5 Thresholding

Binarization is the operation of converting a grayscale image into a binary image whose pixels are logical variables with only one bit - either '0' or '1'. Binarization can be implemented by simply thresholding a grayscale image - assigning '1' to all the pixels whose values are higher than a certain threshold, and assigning '0' to the rest of the pixels.

In Figure 2.7 foreground is dark text and background is bright blank. Figure 2.7 A is produced with a threshold of 127.5. Pixels whose values range from 0 to 127 become binary 0, and pixels whose values range from 128 to 255 become binary 1. In Figure 2.7 B the threshold is manually fine-tuned, in search of a value that could perfectly distinguish text from background. Unfortunately some text is assigned binary 1, while some binary 0s are actually bright background, meaning this perfect threshold does not exist. Otsu's method automatically calculates a threshold that minimizes the intraclass variance [7]. The performance (Figure 2.7 C) is similar to setting the threshold to halfway of the maximal range (Figure 2.7 A). The relatively good result in Figure 2.7 D is given by adaptive thresholding [8], where a pixel is only compared to a statistic (mean, median or Gaussian weighted mean) of a user defined neighbourhood.

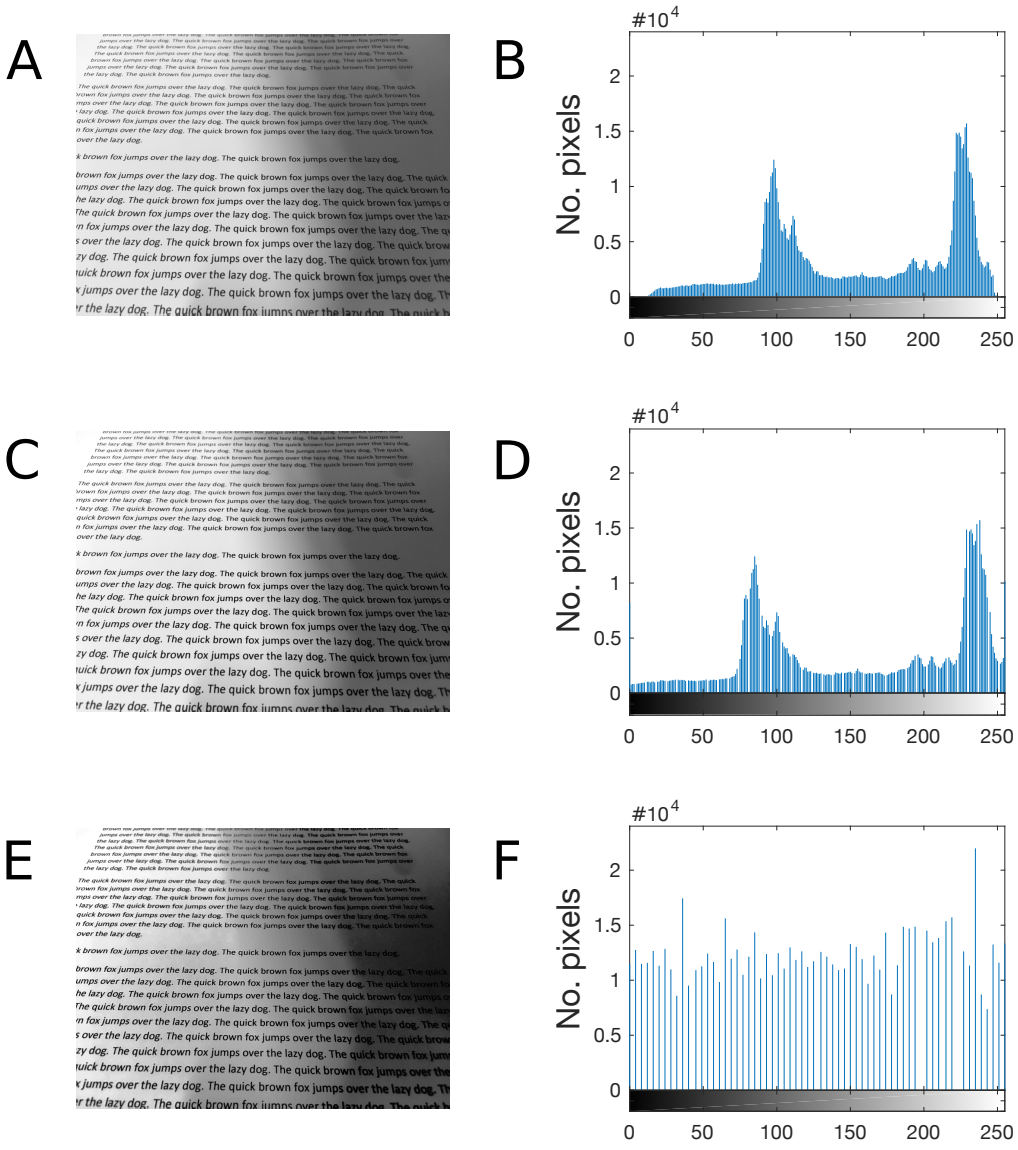


Figure 2.6: Image histograms. (A) Original image of a piece of text under light and shading. (B) Histogram of (A). (C) Result of (A) after curve adjustment. The highest and lowest 1 % pixels are saturated. (D) Histogram of (C). (E) Result of (A) after histogram equalization. (F) Histogram of (E).

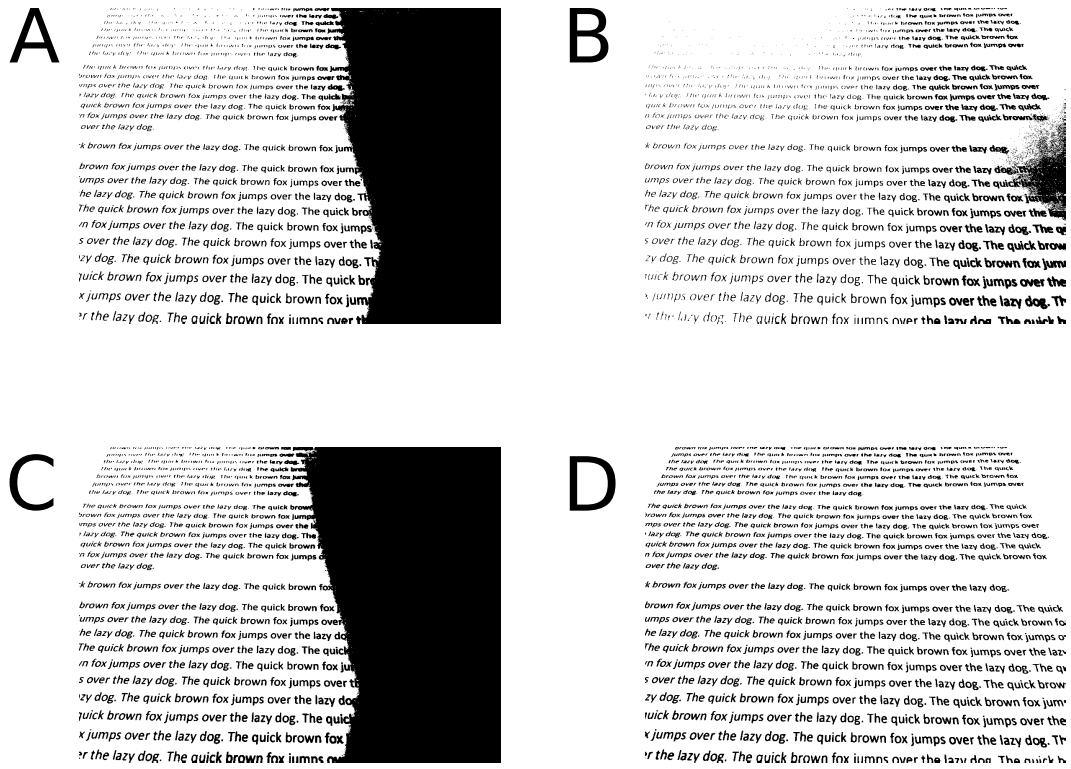


Figure 2.7: Histograms and thresholding. (A) Result of binarization with a threshold in the middle between minimal and maximal value. (B) Result of binarization using a fine-tuned threshold. (C) Result of binarization with a threshold calculated with Otsu's method. (D) Result of binarization using adaptive thresholding.

2.3.6 Morphological operations

Morphological image processing is mainly used on binary images [9, 10]. Morphological operations are able to modify the geometrical properties of images, such as size and shape of foreground objects made up of connected '1's.

Figure 2.8 demonstrates the effects of a few example morphological operations on a binary image (Figure 2.8 A). Simple operations include skeletonizing (Figure 2.8 B, reducing objects to centerlines without changing the structures), area opening (Figure 2.8 C, removing foreground objects smaller than an area threshold), and hole filling (Figure 2.8 D, removing background regions surrounded by foreground). Among the more complicated operations are dilation (Figure 2.8 E) and erosion (Figure 2.8 F), which basically expands and shrinks an object with the help of a structural element - the 'tool' used in these two procedures. Opening (Figure 2.8 G) and closing (Figure 2.8 H) are combinatorial applications of dilation and erosion. In an opening, an erosion is done before a dilation, and in a closing a dilation is done before an erosion.

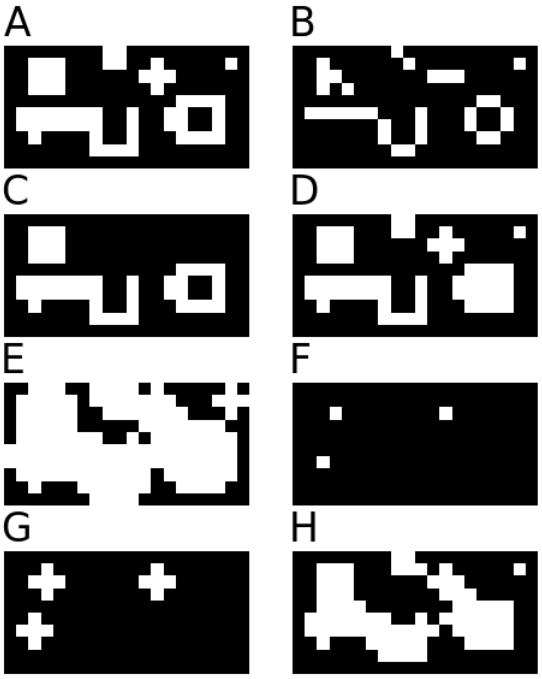


Figure 2.8: Morphological operations on binary images. These images all have the same size of 10×20. (A) Original image. (B) Skeletonizing. (C) Area opening. (D) Hole filling. (E) Dilation. (F) Erosion. (G) Opening. (H) Closing. (E) - (H) Structural element is $[0, 1, 0; 1, 1, 1; 0, 1, 0]$.

2.3.7 Pixel connectivity and connected components

Pixel connectivity and connected components are defined on binary images like Figure 2.8. A pixel's 4-connected neighbors are the pixels on its top, bottom, left and right. A pixel's 8-connected neighbors are the pixels on its top, bottom, left and right, as well as top-left, top-right, bottom-left and bottom-right.

Based on direct 4-connectivity or 8-connectivity, two pixels are indirectly connected if there is a list of pixels where the two are at both ends and each pair of consecutive pixels in the list are connected (4 or 8). A connected component is a collection of pixels in which each pair is directly or indirectly connected. For example, in Figure 2.8 A, B, C, D, E, F, G and H there are 6, 14, 3, 6, 2, 3, 3 and 4 4-connected components, but 6, 6, 3, 6, 1, 3, 3 and 2 8-connected components, respectively.

In fact, connected component is a more abstract and general term in graph theory, which deals with models of graphs made up of vertices and the edges between pairs of vertices [11]. Pixel connectivity is a special case for the 'edges'.

2.4 Microscopy

2.4.1 Light microscopy

In biological research, **microscopy** is used for making minute structures and processes visible to the naked eye (Figure 2.9). **Optical microscopy** or **light microscopy** amplifies, transmits and records visible light signals from the specimen. Alternatives, for example **electron microscopy**, are out of the scope of this text. In this thesis, only light microscopy is discussed.

Figure 2.9: Microscopes in the Museum of Natural History in Berlin, Germany. Usage permission granted.

Apart from directly observing from the eyepiece, it is also common to put a camera at the end of the light path to record the light signal. The recorded data are called micrographs. Analog cameras and photographic films were originally used to capture and store this micrograph. In recent years, CMOS (complementary metal–oxide–semiconductor) and CCD (charge-coupled device), which are able to convert light signal to electronic signal, have been used as sensors by digital microscopes to record micrographs as digital images [12].

Light microscopy technologies can be divided into two categories based on the method they use to illuminate the objects. The first category is **transmitted light microscopy**, including **dark field**, **bright field**, **phase contrast** and **differential interference contrast**. In these microscopes the illumination light passes through the objective and objective lens, and is then amplified and recorded [12]. These technologies are usually used for distinguishing morphological characteristics but not extracting quantitative information about the amount of chemical species.

2.4.2 Fluorescence microscopy

Fluorescence microscopy is used to measure the levels of certain chemical species. In the specimen, each molecule of interest is tagged with a stain, which is usually a fluorophore. The light source of the microscope illuminates the specimen with excitation light of a certain wave length. The fluorophores absorb the excitation light and emit light of a different (usually longer) wavelength. This emission light is captured and recorded by the digital camera in the form of digital images.

Typical fluorescence microscopes include **widefield microscopes**, **confocal laser scanning microscopes**, and **super-resolved microscopes**. Widefield microscopes illuminate the entire field of view and take in light emitted from locations at all depths within a specimen to synthesize a 2D image, while confocal microscopes are able to precisely illuminate only a diffraction-limited focal volume and make sure only light from this volume comes into detector. By scanning through a focal plain or multiple focal plains, confocal microscopes are able to acquire both 2D and 3D images [12].

With both widefield and confocal microscopy, it is difficult to distinguish each single molecule in the acquired images. A basic assumption is that the light intensity of a certain pixel in the recorded image reflects the level of this chemical species at the region represented by this pixel. In this way we are able to infer the relative spatial distribution of the chemical species.

2.4.3 The dimensions of micrographs

With modern digital microscope systems it is possible to image different chemical species / subcellular structures in one experiment. For this purpose, each species of interest needs to be labeled with a different fluorescent marker. The emission light of these fluorescent markers occupy minimally overlapping spectra, and are captured by separate filters accordingly. These separately recorded images will be combined into a single digital image with **multiple channels**. This is comparable to the RGB channels in photography images (Figure 2.3). But there is a difference: in photography images, light of different channels comes from the same source, but in fluorescent microscopy images it usually originates from different chemical species located at the same pixel.

Time lapse microscopy is quite similar to time-lapse photograpy. It images the specimen repetitively with certain time intervals. The resulting digital image will have an additional dimension - time. Time lapse microscopy is widely used in live cell imaging to study biological processes because of its ability to image the specimen at different time points.

In summary, apart from x and y, a micrograph could also have these following dimensions: z (stack), c (channel) and t (time).

2.5 A short note on machine learning

Machine learning [13, 14] is a subset of artificial intelligence. It studies computer programs that carry out various tasks by leveraging statistical data analysis instead of completely relying on hard-coded rules. It is commonly divided into two categories: supervised learning and unsupervised learning.

Supervised learning models are expected to automate tasks that people are already able to do. The procedure of teaching machines how to carry out the task is called 'training', during which machines observe examples of paired input data and output labels. Once they figure out the pattern between them, they gain the ability to label new data independently. Depending on the form of output labels, supervised learning tasks are divided into classification and regression. Classification puts new samples into categories, for example whether a tumour is benign or malignant, whether it will rain tomorrow, whether a picture is of a cat, a dog, or neither. On the other hand, regression assigns new samples with numerical values, for example how long will the patient live, how much will it rain tomorrow, how old is the animal in the picture.

Unsupervised learning models observe only input data without labels. Therefore, they have to figure out the relationship between the samples on their own. Typical unsupervised learning tasks include clustering [15, 16] and low-dimensional data embedding [17–20]. In biology research, they help to determine the proximity between samples and discover new genotypes and phenotypes.

In the past decade deep learning (a subset of machine learning) [21–23] has revolutionized the computer vision field.

2.6 Computer vision

Human beings are able to see objects that reflect, refract or emit light. The light is received by the retina and transformed into neural signals, which are transmitted via optical nerves to the visual cortex of the brain, where the visual information is processed and eventually understood. This process is human vision.

When people see pictures they relate the content to their memories. If they understand the picture they will be able to (1) tell what the picture is of, (2) point at objects they find in the picture and tell what they are, as well as (3) summarize the context of the picture with a sentence. These are analogous to three very common computer vision tasks - image classification, object detection and image captioning.

The purpose of building *in silico* systems emulating human beings' vision is to liberate human beings from repetitive work, and to improve reproducibility. These systems 'see' images and videos fed into them, and work at unparalleled scales and speeds. They are widely used in scenarios such as autonomous vehicles, manufacture and surveillance.

Because micrographs are also digital images and videos, computer vision methods are widely applicable in microscopy image analysis. With the help of the recent advancements in deep learning, this promising field is under fast growth [24, 25]. In the following part of this section, I will discuss several most typical tasks in computer vision, what they are reminiscent of in computational biology, and what applications have already been developed to meet the needs.

2.6.1 On images with single object of interest

Perhaps the simplest type of image is the one with only one main object. If we want to teach a baby to read an image, the first questions could be 'Is it a cat or a dog?', and 'Where is the cat/dog?' even though there is just one animal. Here we talk about images with one main object.

Image classification

As the name suggests, image classification means classifying an image into one of a list of categories. In reality the output of image classification should be a list of probabilities that the image belongs to this category (Figure 2.10 A). Image classification is perhaps the most common, basic and fundamental task in computer vision. It is an essential step in object localization (Figure 2.10 B) and object detection algorithms (Figure 2.11 C), and its models could also

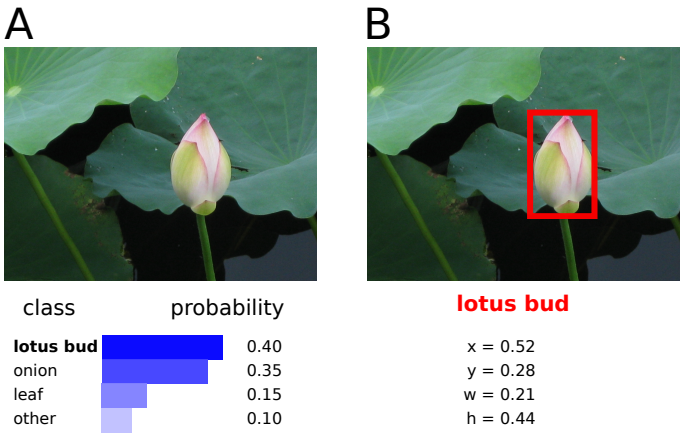


Figure 2.10: Typical tasks in computer vision. (A) Image classification. (B) Object localization.

be reused in semantic segmentation (Figure 2.11 D). The ImageNet Large Scale Visual Recognition Competition (ILSVRC) [26, 27] has benchmarked both object detection and object localization tasks, among other tasks. Over the years, CNN (convolutional neural network) [28] based models, including AlexNet [29], VGGNet [30], GoogLeNet [31], ResNet [32], and EfficientNet [33] have approached and surpassed human level performance in image classification.

Object localization

In object localization, the algorithm is provided with an image and a class label, and is expected to calculate the bounding box of the object belonging to this class (Figure 2.10 B). Because the definition of a bounding box involves 4 numerical values, object localization is often treated as a regression problem. OverFeat [34], winner of the localization task in ILSVRC2013, is the most influential object localization model to date. In OverFeat, an image classifier is trained first, then a bounding box regressor is trained based on the image classifier model.

Classification of single cells and single nuclei

The closest task in microscopy image analysis to (single-object) image classification is probably single cell phenotype classification. In practice, each single cell is represented with either a small image patch including the bounding box of the cell at least, or intensity, morphological and texture descriptors extracted from the small image patch. These tools require users to interactively put single cell image patches into phenotypical categories. After learning from these training

samples the model will be able to classify new data. Examples include CellProfiler Analyst [35], CellClassifier [36], Enhanced CellClassifier [37] and Advanced Cell Classifier [38].

2.6.2 On images with multiple objects of interest

If an image has multiple similarly important objects, it would make more sense to classify it into categories of scenarios rather than categories of objects. It could also be interesting to compose a sentence describing the image. For a person, classifying and describing the content of such images require not only simultaneous recognition of most of the objects, but also the ability of understanding the abstract relationships and interactions between the objects. In computer vision, these two tasks are referred to as **scene classification** and **image captioning** (Figure 2.11 B) [39, 40].

Classification of images with multiple objects is also important in life sciences. In high content screening, microscopy images are classified into different cellular phenotypes, for example different protein subcellular localization patterns [41–43]. Protein subcellular localization classification has also been done for the purpose of basic research using citizen science and CNNs [44]. In computational pathology, histology images are classified into categories like healthy and diseased [45].

2.6.3 Object detection and segmentation

For images containing multiple objects, it is sometimes not enough just to gain a holistic impression and compose a short summary. An image is able to convey much more information than a class label or a sentence.

Semantic segmentation

As we know, an image is a matrix of pixels. Sometimes it is important to know what type of content each pixel belongs to. The process of assigning each pixel to a class is called 'pixel classification' or semantic segmentation (Figure 2.11 D). The typical output of a semantic segmentation task is a new image of the same width and height as the input image. This new image is called the pixel-wise mask. The value of each pixel in the mask encodes the class label of this pixel.

The most important deep learning based semantic segmentation model is the Fully Convolutional Network [46], which achieved state of the art performance on PASCAL Visual Object Classes [47]. For multiple objects belonging to the same class, semantic segmentation models will not distinguish them from each other.

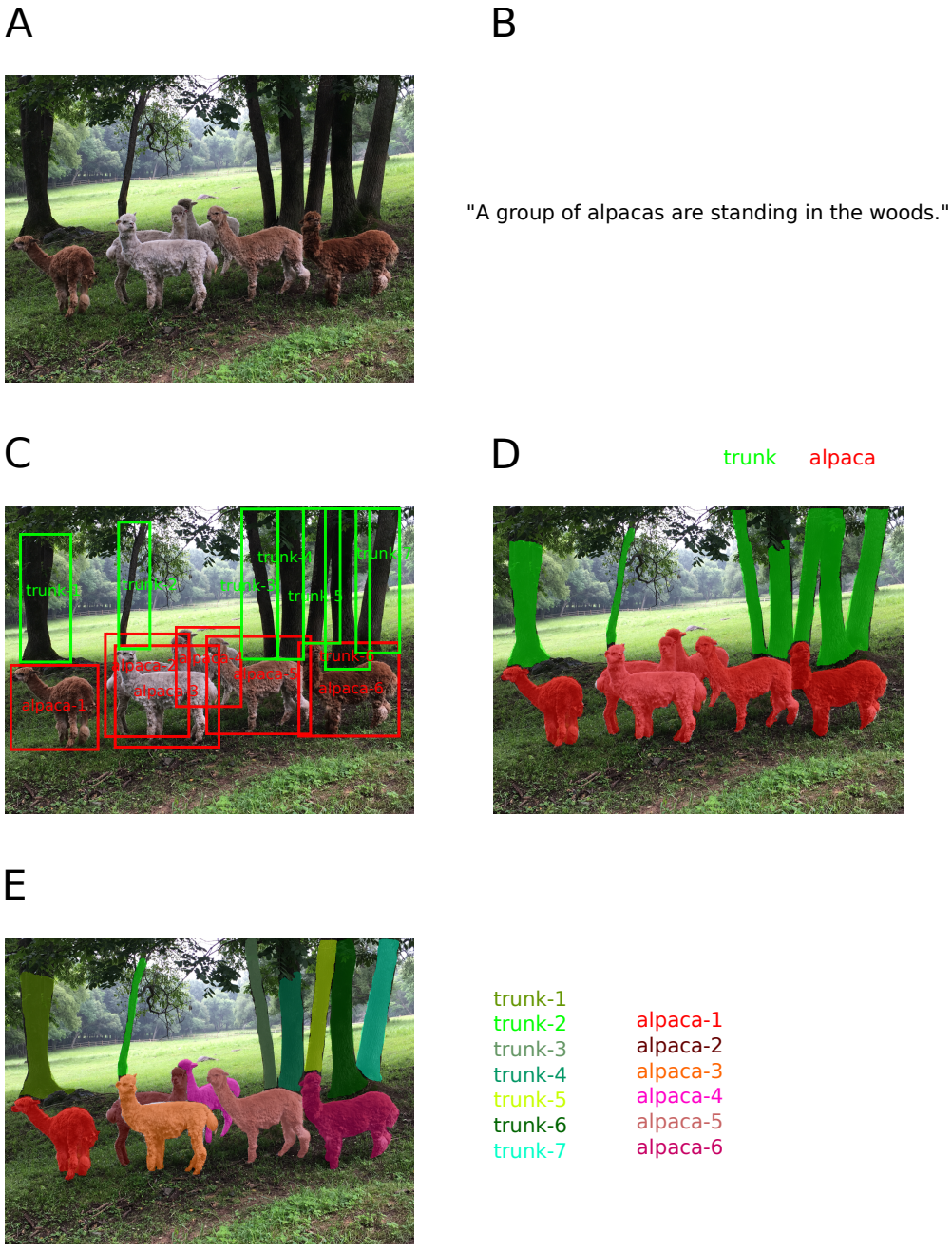


Figure 2.11: Typical tasks in computer vision. (A) Raw image. (B) Image captioning. (C) Object detection. (D) Semantic segmentation. (E) Instance segmentation.

Object detection

Object detection is a task that identifies each individual object even if they are from the same class. Similar to object localization in single object images, object detection produces a bounding box for each object without specifying its pixels. R-CNN is a very important object detection model [48]. It uses region proposal to generate potential bounding boxes and classify them with CNN and SVM (support vector machine). The results will be refined with bounding box regression and duplication detection. Fast R-CNN [49] and Faster R-CNN [50] speed up R-CNN by sharing computation and upgrading the region proposal module. They all achieved state of the art on PASCAL VOC challenges regarding both accuracy and efficiency. Compared to these approaches, YOLO models adopt a simpler pipeline, in which a single neural network predicts bounding box coordinates and object class scores in one evaluation, and is optimized end-to-end. This design makes YOLO models extremely fast [51–53].

Instance segmentation

Among all the typical tasks in computer vision, perhaps the most complicated one is instance segmentation, which requires not only detecting each individual instance of any class, but also identifying the pixels belonging to each of them. For each pixel, instance segmentation methods will assign a class, and an instance index within the class as well. The most widely used model for instance segmentation till now is Mask R-CNN [54], which was built based on Faster RCNN [50]. It achieved better performance than all other single-model entries on instance segmentation.

Detection and segmentation in microscopy image analysis

Segmentation is very useful in biology and medicine because it is often interesting to define contours of nuclei, cells or tissues. Numerous computational approaches have been developed to deal with segmentation of various types of objects [55], such as thresholding [7] and seeded watershed [56]. Here we review a few important tools and frameworks.

ImageJ (FIJI) [57] and CellProfiler [58, 59] are popular biological image analysis platforms. They both support segmentation of foreground objects, which could be cells, nuclei, or subcellular components. ilastik is a segmentation tool that interactively collects user labeled training samples, and classifies each pixel with random forest [60]. DeepCell uses CNNs to classify each pixel by classifying the image patch in which the pixel is the center [61]. FastER interactively trains an SVM to select the region that is most likely to be a cell from a set of generated candidate regions [62].

Based on Fully Convolutional Networks [46], U-Net was built to tackle semantic segmentation problems in biomedical images [4]. It leverages extensive elastic transformation based data augmentation. U-Net won the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks [63], as well as the ISBI Cell Tracking Challenge 2015 on phase contrast and differential interference contrast images [64, 65]. It has been widely adopted, extended to 3D [66], and implemented as ImageJ plugin [67].

Nuclei segmentation and cell segmentation are both instance segmentation tasks, but most of the aforementioned solutions are essentially combinations of semantic segmentation and post-processing. Specifically, after pixels are classified as foreground and background, each connected component (see subsection 2.3.7) of foreground is labeled as an individual instance. In other words, background pixels are used to separate the foreground instances.

The closest task in biological image analysis to object detection is bright spot detection, for example FISH (Fluorescence *in situ* hybridization) signals. Many tools are developed for this task, such as FISH-quant [68, 69], GoIFISH [70] and Aro [71].

2.6.4 Object tracking

Video object tracking

Video object tracking means locating a (moving) object over time [72]. This usually requires first detecting objects in each frame, and then associating objects in consecutive frames. This paradigm is named tracking-by-detection. Important algorithms include Multiple Hypothesis Tracking (MHT) [73,74] and the Joint Probabilistic Data Association Filter (JPDAF) [75, 76]. Simple Online and Realtime Tracking (SORT) [77] is a simple and fast approach that utilizes Kalman filter [78] for motion prediction, and Hungarian algorithm [79] for object association. Its simple and pragmatic design makes it as accurate as and much faster than the state of the art. Its accuracy was later improved by an extension, which integrates object appearance information in addition to location, velocity and bounding box [80].

Object tracking in time-lapse microscopy

Time-lapse microscopy repeatedly images the same biological specimens in one experiment. So it is widely used to observe live single cells, and the single particles inside them, over time. Just like people and vehicles move in videos, cells and particles move in time-lapse microscopy images as well, and this motion makes it not straightforward to find the trajectories of people, vehicles, cells and particles.

In time-lapse microscopy, especially fluorescent time-lapse microscopy, frequent light exposure is often prohibitive for living specimens. This imposes an additional constraint onto the length of time intervals between consecutive frames, and may cause the locations of cells or particles to have largely changed during this interval. This difference makes tracking for time-lapse microscopy more challenging.

Over the years, various cell/nucleus and particle tracking methods have been developed [81, 82]. These methods are categorized into tracking-by-detection approaches based on cell/nucleus/particle detection, tracking-by-segmentation approaches based on cell/nucleus segmentation, and contour evolution (joint segmentation and tracking) approaches [83]. In CTC (Cell Tracking Challenge) [64, 65], which quantitatively evaluates cell/nucleus segmentation and tracking performances, various strategies were adopted by the candidates. Some candidates used contour evolution [83], but the majority adopted different variations of tracking-by-segmentation, such as 'distance-based nearest neighbor linking', 'maximum-overlap-based propagation', and graph-based global optimization. Notably, one of the optimization-based methods used the Viterbi algorithm and achieved superior results in many datasets [84, 85].

Software tools have also been built to benefit end users [86]. For example, tTt [87] allows users to manually and simultaneously detect and track single cells, while automatic tools CellCognition [88], NucliTrack [89] and CellProfiler [58, 59] adopt the tracking-by-segmentation strategy.

The tracking options in CellProfiler cover the most representative variations of tracking-by-segmentation: (1) 'Overlap' selects the object in the previous frame with the greatest amount of spatial overlap with the object under consideration as the predecessor; (2) 'Distance' picks the object in the previous frame with the shortest distance to the object under consideration as the predecessor; (3) 'Measurements' pick the object in the previous frame that is the closest to the object under consideration regarding a calculated measurement as the predecessor; (4) 'Follow Neighbors' assumes objects move in similar directions as their neighbors and formulates tracking as an optimization problem [90]; (5) 'LAP' (linear assignment problem) treats tracking as a global combinatorial optimization problem [91].

These options in CellProfiler are very similar to the strategies used in the video object tracking method SORT [77]. SORT uses distances between objects and overlap between bounding boxes for tracking, and also formulates tracking as an optimization problem. In the improved SORT [80], the appearance of objects in the bounding box was taken into account, just like in CellProfiler the measurements could be used for tracking.

2.7 Live cell imaging

Imaging technologies provide phenotypical measurements of structures like embryos, tissues, cells, and subcellular compartments, relative locations of subcellular components in cells, as well as localization of chemical species within these structures. In short, imaging provides morphological, structural and spatial information.

Live cell imaging studies living cells with time-lapse microscopy. It is able to unveil how the aforementioned morphological, structural and spatial information evolves over time. It offers a unique chance to witness the progress of cellular processes: how embryos develop, how stem cells differentiate, how cells duplicate themselves, how proteins translocate and how genes are transcribed. Based on these observations it might be possible to model the molecular mechanisms, to unveil the causal relationships between phenomena, and to understand the origination and impact of heterogeneity across a cell population and within a cell's life cycle.

Live cell imaging, as well as other time-resolved single cell methods, has advantages over traditional approaches that measure population average snapshots [92–94]. Until now it is the only way to continually monitor individual cells without killing (therefore changing) them, and acquire their histories.

2.7.1 Live cell imaging quantifies cellular dynamics

Technologies that directly acquire the amounts of chemical species, for example blots and sequencing methods, all require the cleavage of cells, therefore cannot evaluate any cell more than once.

On the other hand, fluorescence microscopy is able to sample the levels of certain chemical species in each single cell multiple times without compromising the cells' viability. This is because the levels of the certain chemical species are measured indirectly. Molecules of interest are tagged with fluorophores, which give out emission light under excitation light, and are therefore visible under microscopes.

Thus, the fluorophores act as reporters for the molecules of interest. With conventional widefield epifluorescent microscopy, it is usually hard to distinguish each single molecule in the acquired images. A basic assumption is that the light intensity of a certain pixel in the recorded image reflects the amount of this chemical species at the region (long cube in widefield microscopy) represented by this pixel.

With the help of certain genome editing technologies, cell are able to stably express fluorescently labeled chemical species of interest. With live cell fluorescent labels

and time-lapse microscopy, it is possible to follow certain chemical species in single cells over time. These chemical species of interest could be components of signal transduction pathways, newly transcribed RNA (ribonucleic acid) molecules or even regions of DNA (deoxyribonucleic acid). In the rest of this section I will review a few studies that employed live cell imaging and subsequent image analysis.

2.7.2 Cell signaling

What is cell signaling

Cell signaling or signal transduction is a cellular communication process through which a cell responds to external stimuli that it senses. In this process, a cell's membrane receptors sense physical or chemical signals, and trigger a series of chemical reactions within the cell which finally cause changes on multiple levels, including gene transcription and translation, metabolism and protein post-translational and conformational changes [95].

TGF-β pathway

For example, TGF-β (transforming growth factor β) family members are known to control cell proliferation, cell differentiation, and morphogenesis. Smad proteins are their main intracellular signal transducers [96–98]. At basal state, Smad proteins continuously shuttle between the nucleus and the cytoplasm [97]. TGF-β initiates the signaling process by binding to its receptors, which triggers phosphorylation of receptor Smads (R-Smads, e.g. Smad2, Smad3). Receptor Smads form complexes with the common Smad (Smad4), and push the nucleocytoplasmic shuttling towards the direction of nuclear accumulation [97]. In the nucleus, Smad complexes act together with other transcription factors and regulate gene expression [96, 98].

Studying cell to cell variability in TGF-β response with live cell imaging

Similar to other signaling pathways, TGF-β stimulated translocation of Smad demonstrates variability among genetically identical cells [99,100]. In the following examples, live cell imaging is used to continually monitor the localization change of a Smad protein in large numbers of single cells. Automatic image analysis, including nucleus segmentation, nucleus tracking and molecular signal extraction yield large numbers of individual single cell traces, which made it possible for the authors to conduct meaningful statistical analysis to support their arguments.

Strasen *et al.* [101] tried to dissect the source of this variability in the TGF-β stimulated Smad2 cytoplasmic-nuclear translocation. Using live cell imaging they

acquired quantitative time-resolved measurements of nuclear-cytoplasmic Smad2 ratio. They excluded the effects from cell cycle stage and local cell density. They kept on asking whether the variability originated from the intrinsic stochasticity of chemical reactions or resulted from the heterogeneous cellular state as described by signaling protein levels. They found that sister cells show more similar response profiles than non-sister controls. This indicates that the signaling pathway responds largely deterministically to a heterogeneous cellular signaling protein state.

Frick *et al.* [102] investigated how cells could achieve robust information processing despite the presence of the variability in the response profiles. They imaged live cells and acquired time-series of nuclear Smad3 signal in single cells. They discovered that, although the nuclear Smad3 intensity varied across single cells both before and after TGF-β stimulation, the fold-change of the two is a more precise response to TGF-β stimulation as measured by quartile coefficient of dispersion. The implication is that this 'fold-change detection' mechanism is able to transduce information more robustly under single cell variability.

2.7.3 Cell cycle

What is the cell cycle

The reproduction of cells is fundamental to all life. For single cell organisms it is usually equivalent to the reproduction of the organism. For multicellular organisms it is essential to development and regeneration.

Cell reproduction is a progress made up of the duplication of chromosome and other cellular components, and their distribution into two daughter cells. This process is highly regulated, consisting of many steps, and occurs throughout the entire life cycle of a cell. This life cycle is called the cell cycle [103].

Phases of the cell cycle

Usually the cell cycle is made up of four phases, occuring one after another in the following order: ... G1 - S - G2 - M - G1 ... In G1 phase the cell grows and prepares for DNA synthesis. In S phase DNA replication happens. In G2 phase, the cell gets ready for cell division. G1, S, G2 are collectively named interphase, in which cells prepare for cell division. In M phase, the cell division happens, and this consists of mitosis and cytogenesis, which mean the division of chromosome and the separation of cytoplasm, respectively. Apart from these four phases there is another stage in which cells are not dividing, nor are they preparing for division. This phase is named G0 and is also known as quiescence.

In 2008, FUCCI (fluorescent ubiquitination-based cell-cycle indicator) [104] was invented as reporter for a cell's phase in cell cycle. FUCCI uses GFP (green fluorescent protein)-cdt1 and RFP (red fluorescent protein)-geminin proteins to mark nuclei, which look red in G1 and green in S/G2/M under microscopes.

Studying the regulation of cell cycle arrest with live cell imaging

The cell cycle is regulated positively by cyclin proteins through activation of CDKs (cyclin-dependent kinases) and negatively by CKIs (cyclin-dependent kinases inhibitors) [103, 105]. For example, p21, also known as CKI1, primarily inhibits CDK2. p21 itself is a target of p53, which is activated in response to stresses such as DNA damage. Through p53, p21 and CDK2, DNA damage is able to result in cell cycle arrest or apoptosis [106–108].

In the following examples live cell imaging is used to follow the cell cycle progression of single cells. Automatic image analysis, including nucleus segmentation, nucleus tracking and molecular signal extraction allowed simultaneous recording of both the cell cycle phase and the dynamics of certain proteins (p53, p21 and CDK2) within each individual cell, making it possible to study how the dynamics of those proteins control cell cycle arrest.

It is unknown how cells manage to respond to the sustained DNA damage signals, while at the same time ignoring transient DNA damages that occur during normal growth. Loewer *et al.* [109] used quantitative time-lapse microscopy to monitor the dynamics of p53 and p21 in live cells. They found that only sustained DNA damage was able to cause oscillation of active p53, subsequently trigger p21 expression, and eventually result in cell-cycle arrest or apoptosis.

Spencer *et al.* [110] monitored CDK2 activity in living cells, and found that when proliferating cells exit mitosis, they bifurcate into two different populations. They either immediately build up CDK2 activity so as to start the next cell cycle or suppress CDK2 activity with p21 and enter a transient quiescence state. Again with time-lapse microscopy, Arora *et al.* [111] monitored CDK2 activity and DNA damage. They found that unresolved endogenous replication stress in a mother cell causes daughter cells to enter quiescence via p21 immediately after mitosis. And the lengths of the quiescence of daughters are correlated with the mother's DNA damage. This work explains the choice of whether or not to enter quiescence, which was not explained in the previous work [110]. A similar conclusion was also reached by Barr *et al.* [112].

Reyes *et al.* [113] found that within around one week after ionizing irradiation, a subpopulation of cells sporadically escaped from cell cycle arrest established by

DNA damage. Observation of p53 and p21 showed that these signaling proteins' cell-to-cell variabilities contribute to heterogeneity in the ability of maintaining long cell cycle arrests. Specifically, different oscillation patterns of p53 cause cells to switch between the bistable states in the mutually inhibitive p21-CDK2 relationship, and cell cycle arrest is only maintained when CDK2 activity is low.

Chao *et al.* [114] used FUCCI to observe cell cycle progression, and used computational models to simulation the cell cycle. They concluded that, in early G1 and G2, DNA damage causes an abrupt, all-or-none cell cycle arrest whose duration correlates with the severity of DNA damage. While all of S is comparatively insensitive to DNA damage - increasing DNA damage only leads to slower S progression.

2.7.4 Transcriptional bursting

Observing mRNA (messenger RNA) in live cells is made possible by connecting target genes with MS2 stem loops, which is able to be tagged by MCP (MS2 coat protein)-GFP fusion proteins [115], or with PP7 stem loops and PCP (PP7 coat protein)-GFP [116]. The GFP or other fluorescent component is able to make the mRNA of interest visible under fluorescent microscopy.

The application of MS2 tagging has resulted in the discovery of the bursting nature of gene transcription in both prokaryotic and eukaryotic cells [117,118]. More recently, live imaging of HeLa cells revealed that transcription of HIV (human immunodeficiency virus)-1 RNA is carried out by series of closely spaced polymerases, named convoys, instead of as single isolated enzymes, confirming results from electron microscopy but providing a dynamic view [119]. They also found the HIV-1 promoter exhibit stochastic fluctuations on two different time scales, referred to as 'multi-scale bursting', including a sub-hour scale switch between long permissive and non-permissive periods, and minute scale switch within each transcription convoy. The two switches are regulated separately by different promoter elements. This could be inferred but not confirmed in studies other than live cell imaging.

In the following two works, time-lapse confocal microscopy was used in imaging live *Drosophila* embryos. Like the studies on cell signaling and cell cycle, in these examples nuclei were also automatically segmented and tracked, but the signal to be extracted are not reporter intensities of signaling proteins. Instead, they detected mRNA transcripts and DNA loci such as promoters and enhancers.

Fukaya *et al.* [120] used MS2 and PP7 as reporters to simultaneously visualize the

real-time activities of two linked reporter genes sharing a same enhancer. They found the common enhancer was able to co-activate the two linked reporter genes across transcriptional bursts, meaning the transcriptional bursting of two genes correlated in time. This observation challenged the classical looping model [121] which would suggest an alternating transcriptional bursting pattern, and it also extended the results of 4C assays [122], which could not provide a dynamic or single cell view.

Chen *et al.* [123] simultaneously monitored reporter gene activity, reporter gene promoter location, and the location of a distant enhancer. They observed that productive transcription requires not only physical proximity between enhancer and promoter from insulator pairing, but also an enhanced compaction likely maintained by transcription activity itself. The results argue against the possibility that transient enhancer-promoter proximity is sufficient for sustained transcription, and strongly suggested a causal relationship between transcription and a more confined spatial conformation. Modelling provides new insights on the unclear causal relationship between transcription and topological changes [124, 125].

Chapter 3

eDetect: a fast error detection and correction tool for live cell imaging data analysis

3.1 Introduction

In this chapter, I will describe 'eDetect', a software tool I developed for live cell imaging data analysis [1]. It is suitable for 2D (2-dimensional) fluorescence time-lapse microscopy images. eDetect includes all the necessary components of a typical workflow: nucleus segmentation, nucleus tracking, and molecular reporter signal extraction. But its focus is quality control, whose importance was mentioned in chapter 1 and will be reiterated in detail in this chapter.

eDetect is open source under the MIT license. It is available at `https://github.com/Zi-Lab/eDetect` and `https://sites.google.com/view/edetect/home`.

The content of this chapter is organized as follows. I will first quickly introduce the benchmarking datasets (section 3.2). One of them also serves as demo throughout this chapter. This demo dataset will give readers an impression on what a live cell imaging dataset is like, and how it is analyzed. Following the datasets, I will describe a typical computational data analysis workflow including mainly segmentation and tracking (section 3.3). This prepares the reader for understanding the quantitative evaluation metrics for the computational workflow (section 3.4). With these evaluation metrics and the datasets (section 3.2) we are able to benchmark the automatic analysis workflow (section 3.3) of eDetect and compare it with existing methods (section 3.5), leading to the challenge that all automatic approaches face (section 3.6). To take on the challenge I added error detection and correction modules, which will be described in details in (section 3.7). Then I will present the performance improvement they brought about (section 3.8). In the end, I will report the methods used in eDetect (section 3.9).

3.2 Datasets

Here I present five benchmarking datasets, including four live cell imaging datasets (subsection 3.2.1) used for benchmarking segmentation and tracking, and one high-throughput screening dataset (subsection 3.2.2) used for benchmarking segmentation. One of the live cell imaging datasets will also serve as a demo.

3.2.1 Live cell imaging datasets

First, I present a live cell imaging dataset 'HaCaT-FUCCI', which was cropped from a dataset generated in our lab. The original dataset was designed for studying cell cycle lengths using human HaCaT cells stably expressing CFP (cyan fluorescent protein)-H2B (histone H2B) nuclear marker (Figure 3.1 cyan channel) and mCherry-Geminin (Figure 3.1 red channel) FUCCI cell cycle indicator [104]. 'HaCaT-FUCCI' is available at `https://github.com/Zi-Lab/eDetect/releases`. It will be used throughout this chapter, serving as an example to help make concepts and procedures clear and easy to understand. More details about 'HaCaT-FUCCI' are shown in Table 3.1.

The other three live cell imaging datasets, 'Fluo-N2DH-GOWT1', 'Fluo-N2DL-HeLa' and 'Fluo-N2DH-SIM+' (Table 3.1), are part of the CTC benchmarking datasets [64, 65] and are available at `http://celltrackingchallenge.net/2d-datasets/`. Together with 'HaCaT-FUCCI', they will be used to elucidate the motivation of building eDetect, and evaluate its segmentation and tracking performance.

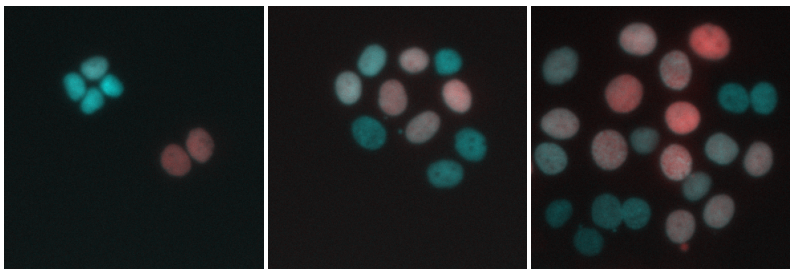


Figure 3.1: Snapshots of HaCaT-FUCCI dataset. Merged CFP-H2B nuclear marker channel (cyan) and mCherry-Geminin FUCCI cell cycle indicator (red) images at the 1st, the 103rd and the 206th (last) frame. Note that these are false color images.

Table 3.1: Live cell imaging datasets used for benchmarking

Dataset name	Fluo-N2DH-GOWT1 [126]	Fluo-N2DL-HeLa [127]	Fluo-N2DH-SIM+ [128]	HaCaT-FUCCI
Cell line	GFP-GOWT1 mouse stem cells	HeLa cells stably expressing H2b-GFP	Simulated nuclei of HL60 cells stained with Hoescht	Human HaCaT cells
Experimentalist	Dr. E. Bártová. Institute of Biophysics, Academy of Sciences of the Czech Republic, Brno, Czech Republic	Mitocheck Consortium	Dr. V. Ulman and Dr. D. Svoboda. Centre for Biomedical Image Analysis (CBIA), Masaryk University, Brno, Czech Republic (Created using MitoGen, part of Cytopacq)	Dr. Guoyu Wu, Max-Planck Institute for Molecular Genetics
Microscope	Leica TCS SP5	Olympus IX81	Zeiss Axiovert 100S with a Micromax 1300-YHS camera	Zeiss Axio Observer.Z1 / 7
Objective lens	Plan-Apochromat 63x/1.4 (oil)	Plan 10x/0.4	Plan-Apochromat 40x/1.3 (oil)	LD Plan-Neofluar 20x/0.4 Korr Ph 2 M27
Pixel size (microns)	0.240 x 0.240	0.645 x 0.645	0.125 x 0.125	0.454 x 0.454
Time step (min)	5	30	29	15

3.2.2 High-throughput screening datasets

'BBBC039' [129] is one of the datasets in the Broad Bioimage Benchmark Collection [130]. It is available at `https://data.broadinstitute.org/bbbc/BBBC039/`.

The dataset is a part of a chemical screen for bioactive compounds. U2OS cells are labeled with Hoechst nuclear stain, which will be used for nucleus segmentation. In this thesis, 'BBBC039' serves the purpose of evaluating the segmentation performances.

3.3 Automatic live cell imaging data analysis

Most of the studies reviewed in section 2.7 used a workflow including nucleus segmentation, nucleus tracking and fluorescence signal quantification. In this section, I will explain nucleus segmentation and nucleus tracking in details using the demo dataset 'HaCaT-FUCCI' from section 3.2.

3.3.1 Input and output

In many live cell imaging applications, we are often interested in how the amount of a certain chemical species changes over time in each single cell. For example, we want to acquire the time dynamics of the aggregated (e.g. median) intensity of mCherry-Geminin (Figure 3.2) in each single cell from dataset 'HaCaT-FUCCI' (Figure 3.1), which has a nuclear marker channel (CFP-H2B, Figure 3.3 top row) and a molecular reporter channel (mCherry-Geminin FUCCI cell cycle indicator, Figure 3.3 bottom row).

The data used in Figure 3.2 A and B are identical - an 18×206 matrix shown in Figure 3.4 B, which is calculated using the matrix in Figure 3.4 A. Each value in Figure 3.4 B is the intensity of the object represented by the corresponding entry of the matrix in Figure 3.4 A. So the two matrices have the same dimensions. Suppose the matrix in Figure 3.4 A is $[A]_{18\times206}$, and the matrix in Figure 3.4 B is $[B]_{18\times206}$. Then the value of A at column x, row y is z ($A_{x,y} = z$), and the value of B at column x, row y is w ($B_{x,y} = w$). Then the zth object in frame x belongs to the yth lineage, and has intensity w.

Now we have raw data (Figure 3.3) and know what the final result looks like (Figures 3.2 and 3.4 B). We need to build a pipeline that converts the former into the latter. I will describe the pipeline in the next subsection.

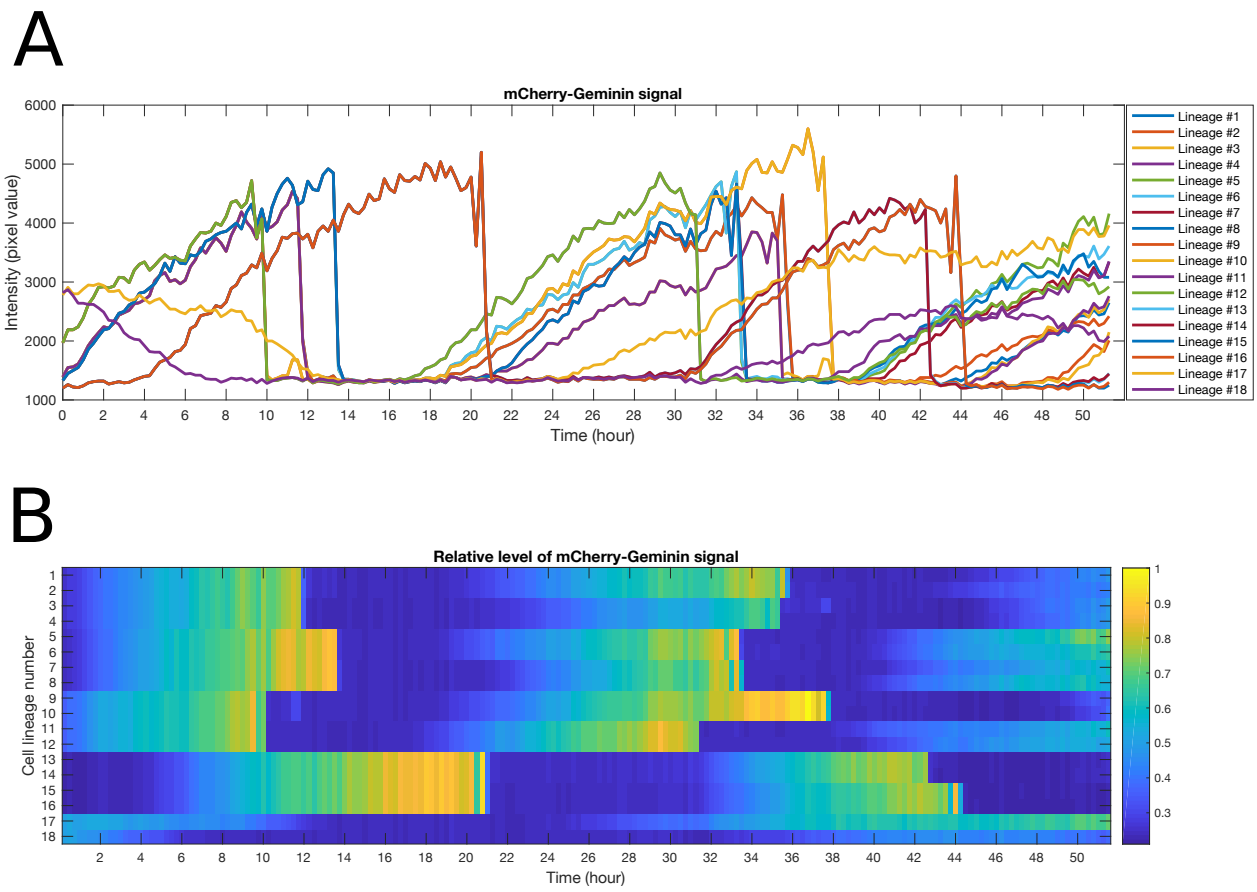


Figure 3.2: Time-series of mCherry-Geminin signal in live single cells. The time-series data is produced from 'HaCaT-FUCCI' dataset. (A) mCherry-Geminin intensities in live single cells presented in a line chart. x-axis: time (hour). y-axis: mCherry-Geminin intensity (grayscale). Each line is a single cell. A line branches when a cell divides. (B) Relative level of mCherry-Geminin intensities in live single cells presented in a heatmap. The heatmap displays a matrix. Each entry is the ratio of the mCherry-Geminin intensity to the original maximum value. Each column represents a frame. Each row represent a single cell in the last frame. For each entry, its predecessor is represented by the entry to its left.

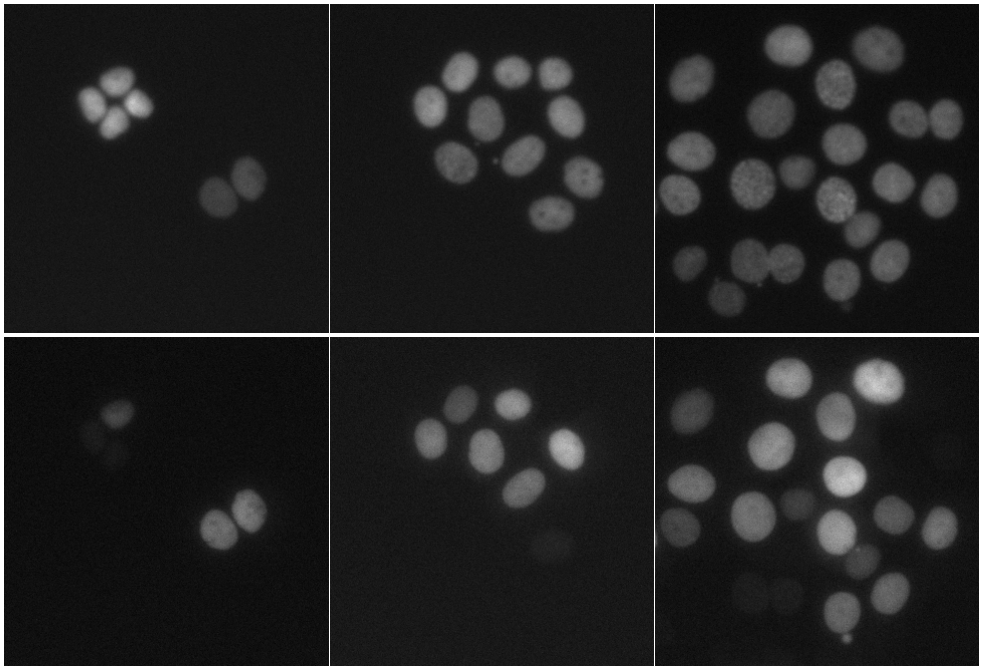
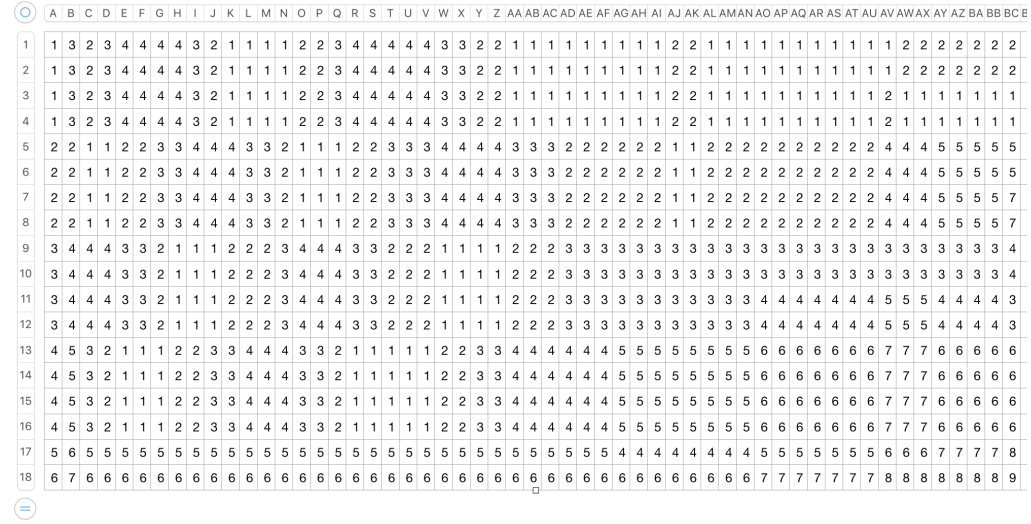
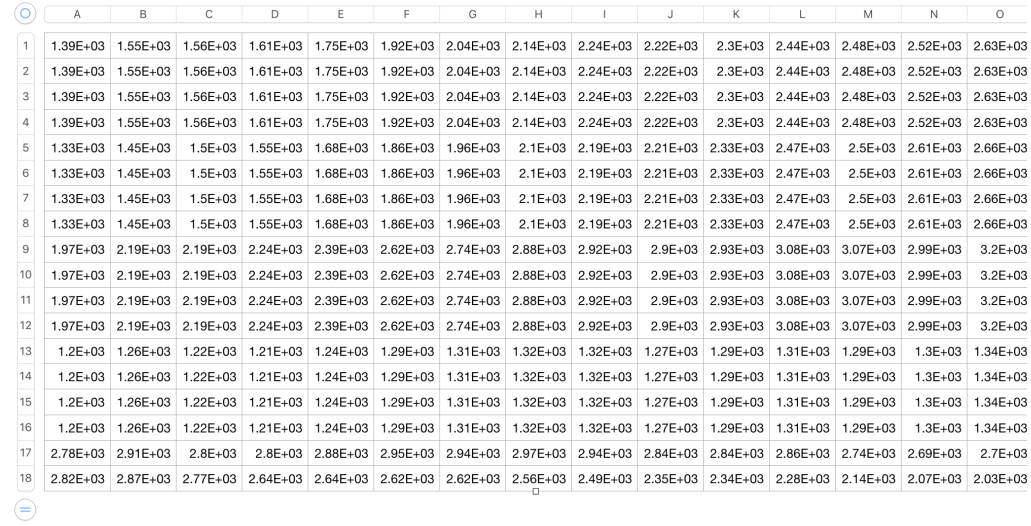


Figure 3.3: Nuclear marker channel and cell cycle indicator channel of dataset HaCaT-FUCCI. Top row: CFP-H2B nuclear marker channel. Bottom row: mCherry-Geminin FUCCI cell cycle indicator channel. Left column: 1st frame. Middle column: 103rd frame. Right column: 206th (last) frame.

A



B



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	1.39E+03	1.55E+03	1.56E+03	1.61E+03	1.75E+03	1.92E+03	2.04E+03	2.14E+03	2.24E+03	2.22E+03	2.3E+03	2.44E+03	2.48E+03	2.52E+03	2.63E+03
2	1.39E+03	1.55E+03	1.56E+03	1.61E+03	1.75E+03	1.92E+03	2.04E+03	2.14E+03	2.24E+03	2.22E+03	2.3E+03	2.44E+03	2.48E+03	2.52E+03	2.63E+03
3	1.39E+03	1.55E+03	1.56E+03	1.61E+03	1.75E+03	1.92E+03	2.04E+03	2.14E+03	2.24E+03	2.22E+03	2.3E+03	2.44E+03	2.48E+03	2.52E+03	2.63E+03
4	1.39E+03	1.55E+03	1.56E+03	1.61E+03	1.75E+03	1.92E+03	2.04E+03	2.14E+03	2.24E+03	2.22E+03	2.3E+03	2.44E+03	2.48E+03	2.52E+03	2.63E+03
5	1.33E+03	1.45E+03	1.5E+03	1.55E+03	1.68E+03	1.86E+03	1.96E+03	2.1E+03	2.19E+03	2.21E+03	2.33E+03	2.47E+03	2.5E+03	2.61E+03	2.66E+03
6	1.33E+03	1.45E+03	1.5E+03	1.55E+03	1.68E+03	1.86E+03	1.96E+03	2.1E+03	2.19E+03	2.21E+03	2.33E+03	2.47E+03	2.5E+03	2.61E+03	2.66E+03
7	1.33E+03	1.45E+03	1.5E+03	1.55E+03	1.68E+03	1.86E+03	1.96E+03	2.1E+03	2.19E+03	2.21E+03	2.33E+03	2.47E+03	2.5E+03	2.61E+03	2.66E+03
8	1.33E+03	1.45E+03	1.5E+03	1.55E+03	1.68E+03	1.86E+03	1.96E+03	2.1E+03	2.19E+03	2.21E+03	2.33E+03	2.47E+03	2.5E+03	2.61E+03	2.66E+03
9	1.97E+03	2.19E+03	2.19E+03	2.24E+03	2.39E+03	2.62E+03	2.74E+03	2.88E+03	2.92E+03	2.9E+03	2.93E+03	3.08E+03	3.07E+03	2.99E+03	3.2E+03
10	1.97E+03	2.19E+03	2.19E+03	2.24E+03	2.39E+03	2.62E+03	2.74E+03	2.88E+03	2.92E+03	2.9E+03	2.93E+03	3.08E+03	3.07E+03	2.99E+03	3.2E+03
11	1.97E+03	2.19E+03	2.19E+03	2.24E+03	2.39E+03	2.62E+03	2.74E+03	2.88E+03	2.92E+03	2.9E+03	2.93E+03	3.08E+03	3.07E+03	2.99E+03	3.2E+03
12	1.97E+03	2.19E+03	2.19E+03	2.24E+03	2.39E+03	2.62E+03	2.74E+03	2.88E+03	2.92E+03	2.9E+03	2.93E+03	3.08E+03	3.07E+03	2.99E+03	3.2E+03
13	1.2E+03	1.26E+03	1.22E+03	1.21E+03	1.24E+03	1.29E+03	1.31E+03	1.32E+03	1.32E+03	1.27E+03	1.29E+03	1.31E+03	1.29E+03	1.3E+03	1.34E+03
14	1.2E+03	1.26E+03	1.22E+03	1.21E+03	1.24E+03	1.29E+03	1.31E+03	1.32E+03	1.32E+03	1.27E+03	1.29E+03	1.31E+03	1.29E+03	1.3E+03	1.34E+03
15	1.2E+03	1.26E+03	1.22E+03	1.21E+03	1.24E+03	1.29E+03	1.31E+03	1.32E+03	1.32E+03	1.27E+03	1.29E+03	1.31E+03	1.29E+03	1.3E+03	1.34E+03
16	1.2E+03	1.26E+03	1.22E+03	1.21E+03	1.24E+03	1.29E+03	1.31E+03	1.32E+03	1.32E+03	1.27E+03	1.29E+03	1.31E+03	1.29E+03	1.3E+03	1.34E+03
17	2.78E+03	2.91E+03	2.8E+03	2.8E+03	2.88E+03	2.95E+03	2.94E+03	2.97E+03	2.94E+03	2.84E+03	2.84E+03	2.86E+03	2.74E+03	2.69E+03	2.7E+03
18	2.82E+03	2.87E+03	2.77E+03	2.64E+03	2.64E+03	2.62E+03	2.62E+03	2.56E+03	2.49E+03	2.35E+03	2.34E+03	2.28E+03	2.14E+03	2.07E+03	2.03E+03

Figure 3.4: Final output of live cell imaging data analysis. (A) Reconstructed cell lineage trees presented in a spreadsheet. Each column is a frame. Each row is a cell lineage. Note that in some columns different entries have the same value. This means they represent the same object, which is the common ancestor of different objects in later frames and therefore belongs to multiple lineages. Only the first 29 columns are shown. (B) Quantified intensity values of objects in the cell lineage trees. This matrix has the same dimensions as the one in A. And each element is the intensity of the object represented by the corresponding entry in A. Only the first 15 columns are shown.

3.3.2 Computational modules

In Figure 3.5 A, the 'nuclear marker' and 'molecular reporter' are images shown in Figure 3.3 A and B, respectively; the 'cell lineage' and 'time-series data' are matrix A and matrix B (Figure 3.4 A and B) in the previous subsection, respectively.

I will describe the pipeline in an analytical manner - start from the result and discuss what is needed for it. Time-series data are acquired by substituting each cell ID in cell lineages with the readout of the object represented by this ID. This operation is **data substitution** (Figure 3.5 A). Cell lineages can be straightforwardly reconstructed from object associations (**lineage reconstruction**, Figure 3.5 A). Object association is the information about who is the predecessor of every object in every frame. The process of identifying the predecessors is **object tracking** (section 2.6 and Figure 3.5 A). In turn, knowing the predecessor of every object in every frame implies knowing every object in every frame, which is the result of object identification or **object detection** (section 2.6, Figure 3.5 A). In object detection, object locations are derived from nuclear marker images. An object readout is calculated by aggregating the pixel values in the quantification region (segmented nucleus or cytoplasm) of the object in the molecular reporter channel, using a single statistic, usually median or mean. This process is quantification or **data aggregation** (Figure 3.5 A). The regions in which the pixels are to be aggregated are acquired by **object segmentation** (section 2.6 and Figure 3.5 A). Object segmentation also uses nuclear marker images, but it not only identifies each object but also specifies their contours, which are then adapted to the molecular reporter channel for data aggregation. This workflow takes input from raw data, transforms them to intermediate results using computational modules, and in the end produces final results (Figure 3.5 A).

This is an exhaustive but abstract pipeline. In practice, software tools may adopt its variations. In tTt and qTfy [87], detection is merged into manual tracking (Figure 3.5 B), and there is also no explicit lineage reconstruction step. So tracking produces object location, object association and lineages. In eDetect [1] there is also no separate object detection step (Figure 3.5 C). Detection is done by segmentation. This also holds true with CellProfiler [58, 59] and NucliTrack [89], but in NucliTrack there is no explicit lineage reconstruction, and in CellProfiler does not support lineage reconstruction after tracking.

In these pipelines, the most challenging steps for automatic analysis are segmentation and tracking (Figure 3.6). In the next section I will review the metrics used for benchmarking these two tasks.

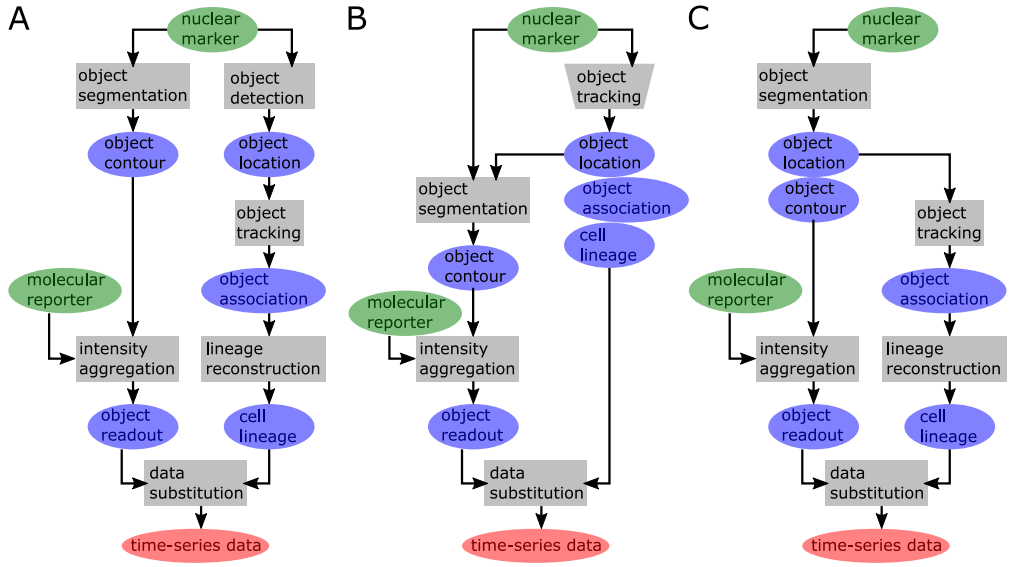


Figure 3.5: Typical live cell imaging data analysis pipelines. Green ellipses: raw input data sources (images). Grey rectangles: automatic analysis modules. Grey trapezoids: manual analysis modules. Blue ellipses: intermediate results. Red ellipses: final output. Arrows represent dependency. The concept at the head of an arrow depends on the concept at the tail of the arrow. Data pointing to a module are input data of the module. Data that a module points to are output data of the module. (A) An abstract workflow. (B) tTt/qTfy's implementation of the abstract workflow. Object detection is merged into object tracking, which is done manually. There is no separate lineage reconstruction. (C) eDetect's implementation of the abstract workflow. Object detection is merged into object segmentation.

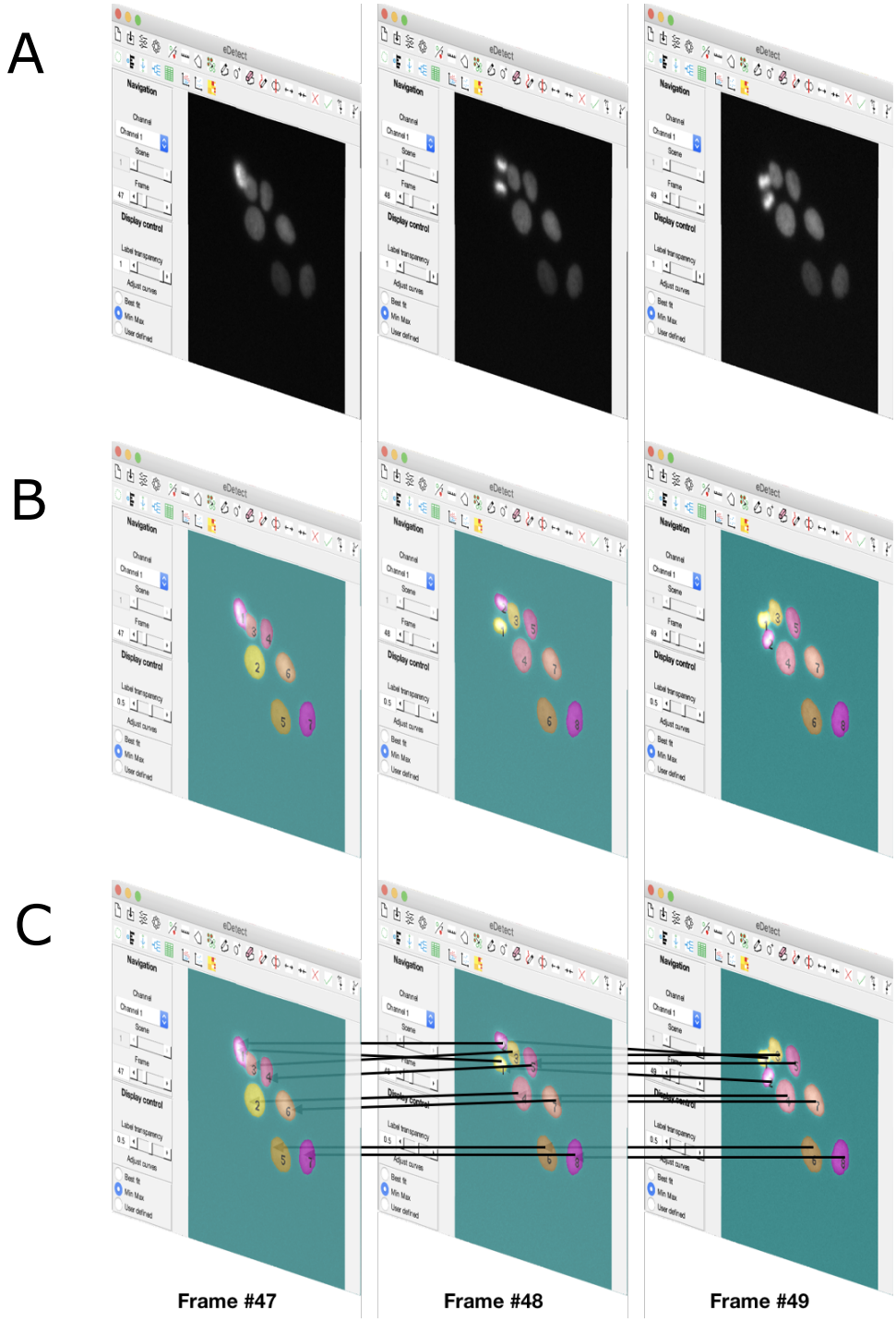


Figure 3.6: Segmentation and tracking using nuclear marker channel. (A) Nuclear marker channel images of Frames 47 to 49. (B) Nuclear marker channel images of Frames 47 to 49 together with segmentation results. Segmentation results are shown as semi-transparent masks superimposed onto the segmented nuclei. (C) Nuclear marker channel images of Frames 47 to 49 together with segmentation and tracking results. Tracking results (object associations) are shown as arrows. For each arrow, the object at its head is the predecessor of the object at its tail.

3.4 Benchmarking

Benchmarking a computational method means evaluating the similarity between its Pr (prediction) and the correct answer, which is usually called reference or GT (ground truth), using a similarity index. Ground truth is often generated by manual annotation. The ground truth and the prediction are in the same format.

The CTC [64, 65] benchmarks candidate algorithms based on two main metrics, namely SEG (segmentation accuracy) and TRA (tracking accuracy) [131], as well as a few additional metrics including the CT (complete track) [132]. Based on CT we created a related index, the CL (complete lineage). eDetect will be benchmarked using SEG, TRA, as well as F_1 scores of CT, CL, and segmentation. Executables implementing SEG and TRA are provided by CTC. CT is implemented according to refs [64, 65].

3.4.1 Examples of segmentation and tracking errors

The errors made in segmentation could be roughly categorized into four groups: (1) false negative - algorithm fails to detect a nucleus (Figure 3.7 A), (2) false positive - algorithm detects an object that is not a nucleus (Figure 3.7 B), (3) over-segmentation - a nucleus is divided into multiple objects (Figure 3.7 C), and (4) under-segmentation - multiple nuclei are identified as one object (Figure 3.7 D). In addition, inaccurate segmentation also affects the quality of the data. An inaccurately segmented object is correctly identified / detected (no false negative, false positive, over-segmentation or under-segmentation with the object), but the object contour is different from the ground truth.

Tracking errors are the cases in which the predecessor of an object is assigned incorrectly. An example of a tracking error is shown in Figure 3.7 E - F: the algorithm thinks the predecessor of 49.1 is 48.2, but in fact, the predecessor of 49.1 is 48.1 (and the predecessor of 49.2 is 48.2).

3.4.2 The Jaccard similarity index

The Jaccard similarity index (or the Jaccard similarity coefficient) is a metric that measures the similarity between two sets by calculating the ratio between the cardinality of their intersection and the cardinality of their union [133]. The Jaccard similarity index is also named IoU (intersection over union). The Jaccard index (J) of two sets A and B is calculated as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{3.1}$$

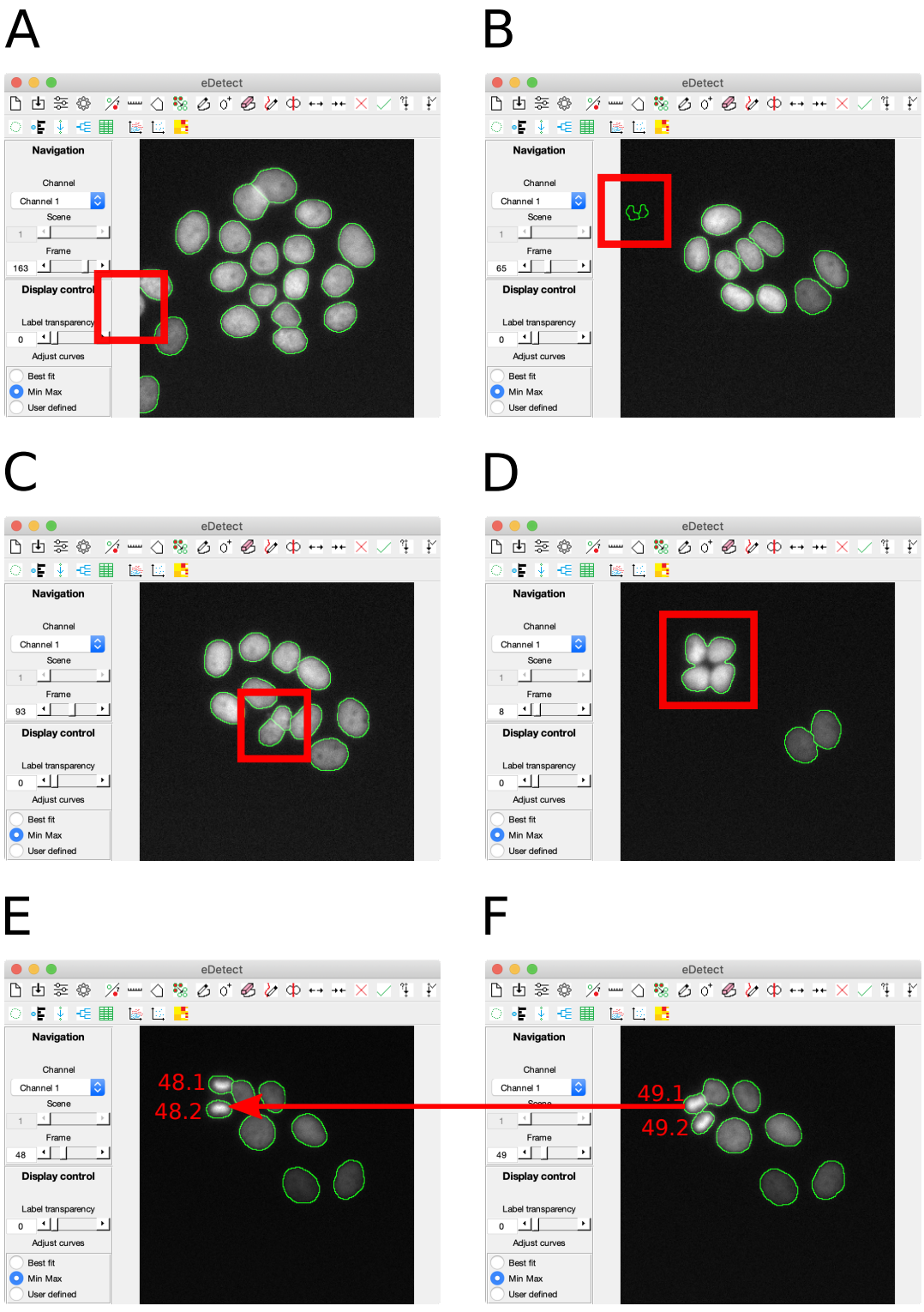


Figure 3.7: Examples of segmentation errors. Each example represents a type of segmentation error. (A) False negative. (B) False positive. (C) Over-segmentation. (D) Under-segmentation. (E)-(F) Tracking error.

in which $A \cap B$ means the intersection of sets A and B, $A \cup B$ means the union of sets A and B, and $|X|$ means the cardinality (number of elements) of set X.

3.4.3 Segmentation accuracy

The SEG of a ground truth segmentation (S_{GT}) equals the Jaccard similarity index between the ground truth segmentation and the predicted segmentation (S_{Pr}) that matches with the ground truth segmentation [64, 65]:

$$J(S_{Pr}, S_{GT}) = \frac{|S_{Pr} \cap S_{GT}|}{|S_{Pr} \cup S_{GT}|}, \tag{3.2}$$

in which both S_{Pr} and S_{GT} are sets of pixel coordinates in an image. A predicted segmentation (S_{Pr}) is defined to match the ground truth segmentation (S_{GT}) if and only if

$$|S_{Pr} \cap S_{GT}| > 0.5|S_{GT}|. \tag{3.3}$$

By this definition, each ground truth segmentation S_{GT} could match with at most one predicted segmentation S_{Pr}. The SEG of a video is the mean of the SEGs of all GT objects in the video.

For example, in Figure 3.8 the S_{GT} has 20 pixels and the S_{Pr} has 27 pixels. Their intersection has 16 pixels and their union has 31 pixels. Therefore $|S_{Pr} \cap S_{GT}| = 16 > 10 = 0.5|S_{GT}|$, meaning they match, and the Jaccard similarity index is $\frac{16}{31}$.

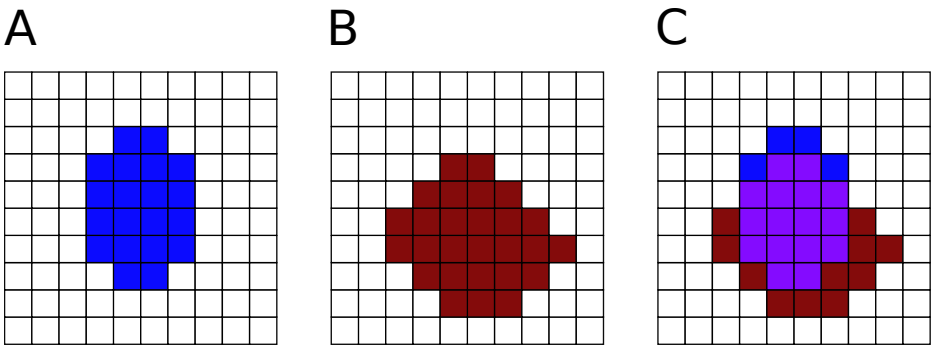


Figure 3.8: Jaccard similarity index used as segmentation accuracy. (A) Ground truth segmentation (S_{GT}) is marked in blue. (B) Predicted segmentation (S_{Pr}) is marked in red. (C) Intersection ($S_{Pr} \cap S_{GT}$ purple) and union ($S_{Pr} \cup S_{GT}$ purple or blue or red) of ground truth and prediction. $|S_{Pr} \cup S_{GT}| = 31$ and $|S_{Pr} \cap S_{GT}| = 16$ so IoU or Jaccard index is $\frac{16}{31}$. And $|S_{Pr} \cap S_{GT}| = 16 > 10 = 0.5|S_{GT}|$.

3.4.4 Tracking accuracy

Both the tracking result (predicted) and the reference (ground truth) will be converted into the form of acyclic oriented graphs (AOG), which are trees capturing the genealogy of the tracked single cells (Figure 3.9 left) [131]. Acyclic Oriented Graph Matching (AOGM) measures the difficulty of modifying one AOG into another AOG, and the tracking accuracy is calculated as [64, 65]:

$$TRA = 1 - \frac{AOGM(AOG_{GT}, AOG_{Pr}) - AOGM(AOG_{GT}, AOG_0)}{AOGM(AOG_{GT}, AOG_0)}, \tag{3.4}$$

in which AOG_{GT} is the AOG of ground truth, AOG_{Pr} is the AOG of prediction made by tracking algorithms, and AOG_0 is the AOG representing a null graph.

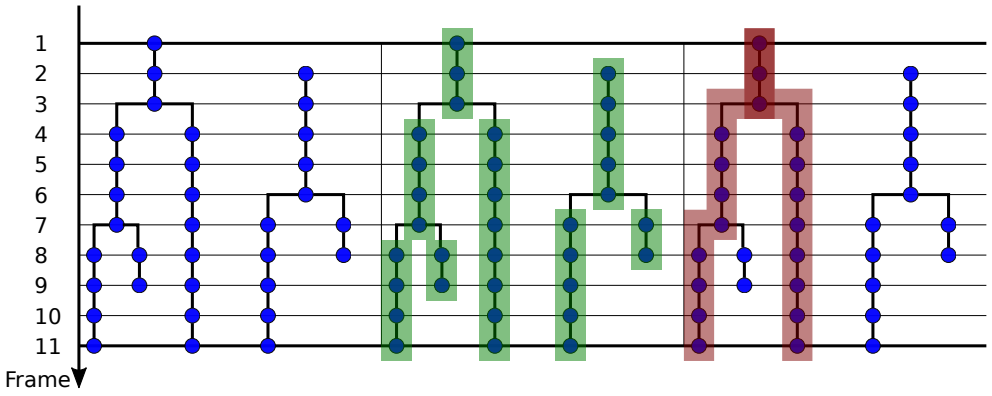


Figure 3.9: Complete tracks and complete lineages. Schematic representation of typical cell lineage trees (left), their complete tracks (middle) and their complete lineages (right). The vertical axis indicates the frames of the time-lapse video. Each blue circle represents an object (cell or nucleus). A line segment connecting two circles means the object above is the predecessor of the object below. Each green rectangle represents a complete track. Each red polygon represents a complete lineage.

3.4.5 Complete tracks and complete lineages

In time-lapse videos, each cell / nucleus could (and probably will) be imaged more than once, therefore corresponds to several objects, each appearing in several consecutive frames. A cell lineage tree, either predicted or annotated as ground truth, is made up of associated objects (Figure 3.9 left). If two objects are associated, the one above is the predecessor of the one below and the one below is the successor

of the one above. If one object has only one successor, then the two of them are one cell / nucleus imaged in two consecutive frames. If one object is the common predecessor of two objects, this means a mother cell (the common predecessor) divides into two daughter cells (the two successors).

The definition of a CT [64, 65, 132] is a group of connected objects that represent one identical cell (Figure 3.9 middle). When a cell divides, neither of the two daughters belongs to the same CT as the mother cell does. Also the two sisters each has its own CT. CT, together with a few other metrics mentioned in Ulman *et al.* [65], is considered to be more biologically meaningful than TRA.

Based on CT, a CL is defined as a complete and uninterrupted genealogy that spans the entire video. Each CL is a unique cell in the last frame of the video, together with its entire history since the first frame of the video (Figure 3.9 right). Any object in any frame other than the last may belong to multiple CLs, if multiple objects in the last frame are its descendants. In other words, CLs may overlap with each other. For example, each row in Figure 3.4 A represents a CL.

3.4.6 The F_1 score

In binary classification between positive (P) and negative (N) classes, TP (true positive) is the number of positive samples correctly predicted to be positive, TN (true negative) is the number of negative samples correctly predicted to be negative, FP (false positive) is the number of negative samples incorrectly predicted to be positive, and FN (false negative) is the number of positive samples incorrectly predicted to be negative. Precision is the proportion of TP out of all positive predictions, and recall is the proportion of TP out of all positive samples:

$$precision = \frac{TP}{TP + FP}, \tag{3.5}$$

$$recall = \frac{TP}{TP + FN}. \tag{3.6}$$

F_1 score is the harmonic mean of precision and recall [134]:

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2}{\frac{TP+FN}{TP} + \frac{TP+FP}{TP}} = \frac{2TP}{2TP + FP + FN}. \tag{3.7}$$

3.4.7 F_1 scores of complete tracks and complete lineages

The F_1 score is used to evaluate the correctness of calculated CTs and CLs:

$$F_1(CT) = \frac{2|CT_{TP}|}{2|CT_{TP}| + |CT_{FP}| + |CT_{FN}|} = \frac{2|CT_{GT,Pr}|}{|CT_{GT}| + |CT_{Pr}|}. \tag{3.8}$$

$$F_1(CL) = \frac{2|CL_{TP}|}{2|CL_{TP}| + |CL_{FP}| + |CL_{FN}|} = \frac{2|CL_{GT,Pr}|}{|CL_{GT}| + |CL_{Pr}|}. \quad (3.9)$$

In these equations, $|CT_{GT}|$ and $|CT_{Pr}|$ mean numbers of ground truth and predicted CTs, respectively, while $|CL_{GT}|$ and $|CL_{Pr}|$ mean numbers of ground truth and predicted CLs, respectively. $|CT_{GT,Pr}|$ means the number of ground truth CTs that are also correctly predicted. $|CL_{GT,Pr}|$ means the number of ground truth CLs that are also correctly predicted. A ground truth CT_{GT} is considered correctly predicted if there is a predicted CT_{Pr} that expands exactly the same range of frames as CT_{GT}, and in each of these frames the two objects from each CT match (as defined in subsection 3.4.3). Similarly, a ground truth CL_{GT} is considered correctly predicted if there is predicted CL_{Pr} that expands exactly the same range of frames as the CL_{GT} (which always holds true by definition), and in each of these frames the two objects from each CL match (as defined in subsection 3.4.3). Note that in Ulman *et al.* [65] the term 'CT' is used to represent the meaning of $F_1(CT)$.

3.4.8 F_1 scores of segmentation

F_1 score is also considered a more biologically relevant segmentation metric than SEG [129]. F_1 scores of segmentation is only affected when objects' identities are wrong, for example in cases of false negatives, false positives, over-segmentation and under-segmentation (Figure 3.7). While SEG is affected as long as any S_{GT} and its 'matched' S_{Pr} do not have identical contours.

The F_1 score of an image is calculated as follows: suppose there are N_{GT} ground truth segmentation objects (S_{GT}s) and N_{Pr} predicted segmentation objects (S_{Pr}s). The IoU threshold is T_{IoU}, which is usually set to be larger than 0.5. Now we construct a matrix with N_{GT} rows and N_{Pr} columns: $[M]_{N_{GT} \times N_{Pr}}$. For any element in this matrix, let $M_{n_{GT},n_{Pr}} = 1$ if and only if the IoU between the n_{GT}th S_{GT} and the n_{Pr}th S_{Pr} is larger than the threshold T_{IoU}, otherwise let $M_{n_{GT},n_{Pr}} = 0$. Because $T_{IoU} > 0.5$, there will not be more than one '1' in any column or any row. Once we get the matrix, TP equals the total number of '1's, FP equals the total number of all-0 columns, and FN equals the total number of all-0 rows. Then F_1 is calculated using TP, FP, and FN. The F_1 score of a dataset is the mean of F_1 scores of all images [129].

F_1 score of segmentation will be used to evaluate segmentation performance on the 'BBBC039' dataset (section 3.2) [129].

3.5 Performance of automatic data analysis

3.5.1 Performance on live cell imaging datasets

The automatic segmentation and tracking functions of eDetect were benchmarked on the metrics SEG, TRA, $F_1(CT)$ and $F_1(CL)$ using the live cell imaging datasets 'HaCaT-FUCCI' and three datasets from the CTC subsection 3.2.1. Its performances are compared with those of CellProfiler [58,59] and the highest three scores from the challenge candidates. Time consumptions of eDetect and CellProfiler were also recorded to evaluate their efficiencies.

In Tables 3.2, 3.3, 3.4 and 3.5, 'CellProfiler' means the results of CellProfiler automatic analysis, and 'eDetect' means the results of eDetect automatic analysis. 'CTC 1st', 'CTC 2nd' and 'CTC 3rd' each means the highest, second highest and third highest scores from all CTC candidates.

eDetect's performances on SEG and TRA are acceptable. This is evidenced by the fact that eDetect is consistently better than CellProfiler in SEG (Table 3.2) and is comparable with CellProfiler in TRA (Table 3.3). eDetect is also comparable with the highest scores of challenge candidates in SEG and TRA only except with the 2nd video of 'Fluo-N2DH-SIM+' (Table 3.2 and Table 3.3).

However, when it comes to $F_1(CT)$, both eDetect and CellProfiler are worse than the highest scores of challenge candidates (Table 3.4). On $F_1(CL)$ eDetect is consistently better than or as good as CellProfiler, but is almost consistently worse than the highest scores of challenge candidates only except with the 2nd video of 'Fluo-N2DL-HeLa' (Table 3.5).

In summary, eDetect is overall more accurate than CellProfiler on SEG ('Mean' in Table Table 3.2) and $F_1(CL)$ ('Mean' in Table 3.5), and comparable to CellProfiler on TRA ('Mean' in Table 3.3) and $F_1(CT)$ ('Mean' in Table 3.4). eDetect is also consistently much faster than CellProfiler (Table 3.6). However, eDetect is overall comparable to the highest scores from CTC only on SEG ('Mean' in Table Table 3.2), and worse than the highest scores from CTC on TRA ('Mean' in Table 3.3), $F_1(CT)$ ('Mean' in Table 3.4) and $F_1(CL)$ ('Mean' in Table 3.5). Note that the highest scores of each CTC video are ranked individually, so the highest scores are contributed by various teams. In other words, eDetect and CellProfiler are compared with the winners on each separate video.

Table 3.2: Benchmarking automatic segmentation algorithms on SEG

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.6826	0.8300	0.7256	0.7794	0.8287	0.2584	0.6841	0.8283
eDetect	0.7449	0.9377	0.8065	0.8473	0.8364	0.5373	0.7850	0.9837
CTC 1st	0.7724	0.9413	0.8270	0.8469	0.8652	0.6577	0.8184	-
CTC 2nd	0.7262	0.9140	0.8251	0.8457	0.8640	0.6406	0.8026	-
CTC 3rd	0.7221	0.8980	0.7774	0.8140	0.8508	0.6298	0.7820	-

Table 3.3: Benchmarking automatic tracking algorithms on TRA

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.9470	0.9467	0.8982	0.9244	0.9727	0.3638	0.8421	0.9695
eDetect	0.9279	0.9465	0.9707	0.9682	0.9817	0.3249	0.8533	0.9870
CTC 1st	0.9654	0.9746	0.9858	0.9794	0.9926	0.9501	0.9747	-
CTC 2nd	0.9617	0.9578	0.9804	0.9665	0.9916	0.9212	0.9632	-
CTC 3rd	0.9597	0.9532	0.9762	0.9620	0.9876	0.8918	0.9551	-

Table 3.4: Benchmarking automatic tracking algorithms on $F_1(CT)$

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.0541	0.1881	0.0556	0.1753	0.3122	0.0057	0.1318	0.0252
eDetect	0.0237	0.0245	0.3208	0.1271	0.3596	0.0006	0.1427	0.2482
CTC 1st	0.7547	0.5984	0.4678	0.4106	0.7500	0.0878	0.5116	-
CTC 2nd	0.6667	0.5918	0.4517	0.3243	0.6878	0.0870	0.4682	-
CTC 3rd	0.5938	0.5652	0.3053	0.2397	0.5376	0.0695	0.3852	-

Table 3.5: Benchmarking automatic tracking algorithms on $F_1(CL)$

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.7317	0.6316	0.4809	0.3973	0.4444	0.0000	0.4477	0.0541
eDetect	0.9048	0.9032	0.7164	0.6007	0.4524	0.0000	0.5963	0.1081
CTC 1st	0.9743	0.9655	0.7778	0.5908	0.9882	0.5684	0.8108	-
CTC 2nd	0.9500	0.9630	0.5258	0.4361	0.9762	0.1887	0.6733	-
CTC 3rd	0.9474	0.9286	0.3046	0.3149	0.9639	0.1818	0.6069	-

Table 3.6: Time consumption of automatic analysis

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	27min06s	26min53s	5min09s	5min14s	2min41s	10min41s	13min	4min08s
eDetect	30s	28s	54s	3min00s	19s	47s	1min	24s

3.5.2 Performance on high-throughput screening datasets

eDetect's automatic segmentation was also separately benchmarked using the dataset 'BBBC039' subsection 3.2.2 and F_1 score of segmentation subsection 3.4.8, which is considered more biologically relevant than SEG [129].

As we see in Figure 3.10, eDetect performs consistently better than CellProfiler when the IoU threshold (T_{IoU}) varies.

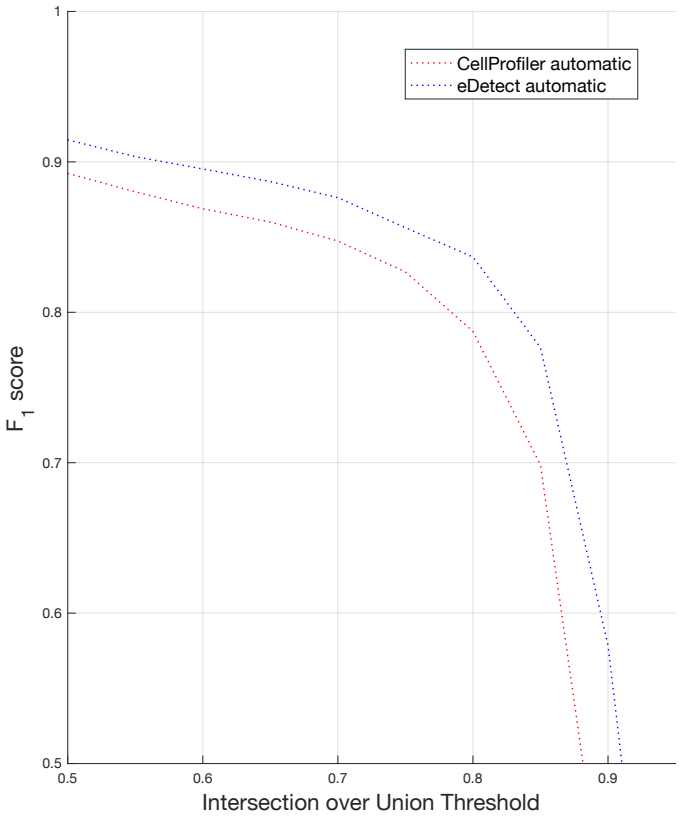


Figure 3.10: Comparing eDetect and CellProfiler based on F_1 scores of segmentation. x-axis: the minimal IoU (Jaccard index) to consider two segmented objects to be the same nucleus. y-axis: F_1 score of segmentation. Red dotted line: performance of CellProfiler. Blue dotted line: performance of eDetect.

3.6 Challenges in live cell imaging data analysis

3.6.1 Automatic algorithms are necessary for large datasets

Automatic data analysis has an obvious merit: it scales much better than manual analysis. With small datasets, for example those with few cells [119], automatic segmentation and tracking are not clearly preferable. But automated microscopes are able to generate voluminous time-lapse videos recording large number of cells for long times. In some studies, tens of thousands of cells [102] or even more [101] are analyzed. In this situation manual analysis it is simply prohibitive.

As an example, I tried analyzing 'HaCaT-FUCCI' with tTt and qTfy [87], which is designed for manual tracking and automatic segmentation using tracking results (Figure 3.5). In 'HaCaT-FUCCI', 6 cells become 18 within 206 frames, yielding about 2314 objects in total - a number much lower than some studies [101, 102]. Still, manual tracking took around 40 minutes and manual correction time of segmentation was around 150 minutes. Complete manual segmentation will probably take longer. Automatic analysis, on the other hand, took about 4 minutes with CellProfiler and less than half a minute with eDetect (Table 3.6).

3.6.2 Automatic analysis makes mistakes

Unfortunately, automatic algorithms cannot guarantee 100 percent accuracy, as it has never been achieved in CTC by any participating team by the time the papers were published [64, 65]. In addition, the performances of eDetect and CellProfiler on the live cell imaging datasets agreed with the conclusion (Tables 3.2, 3.3).

3.6.3 Quality control is important

For a complex computational pipeline, the final results can be highly susceptible to upstream errors. In addition, the dissimilarity in formats between raw input and final output could make upstream errors difficult to spot by simply looking at final results. Thus, upstream errors may silently introduce noise whose damage is hard to estimate. This holds true for many biological image analysis pipelines which often start from segmentation [129], such as live cell imaging data analysis.

Live cell imaging studies, especially long term ones, are particularly sensitive to errors. This is because live cell imaging studies require complete cell lineages (CL in section 3.4) to arrive at valid biological conclusions. Even extremely low error rates in individual images/instances will be amplified in the long time-course data [135], because a cell lineage is correctly reconstructed if and only if all objects are correctly segmented and all their predecessors are correctly assigned.

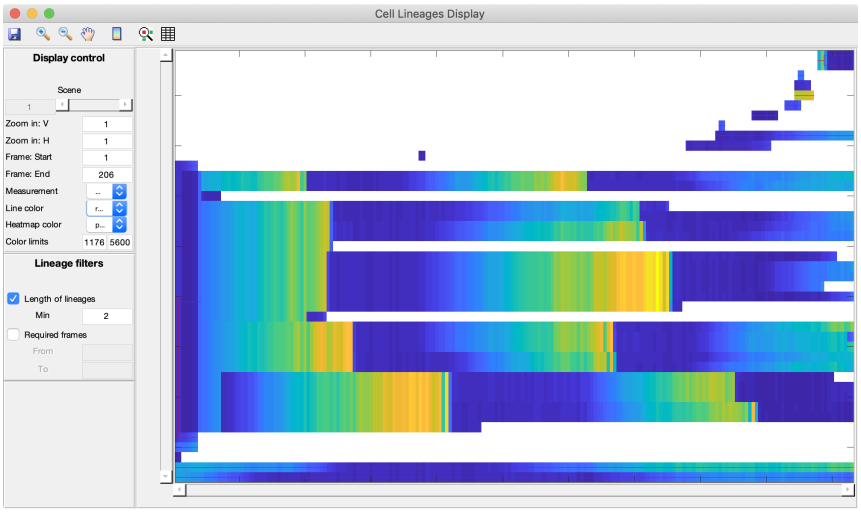


Figure 3.11: Time-series data acquired from eDetect automatic analysis is displayed in heatmap. The original dataset is 'HaCaT-FUCCI'. The color of the heatmap entries encodes the time-series of mCherry-Geminin nuclei intensity. The structure of the heatmap represents the cell lineages.

For example, in 'HaCaT-FUCCI' dataset there are six cells in the first frame. During the course of imaging four of them divided twice, yielding 18 cells in the last (206th) frame. In these six lineages that lasted from the first frame to the last frame (Figure 3.2 B), 2314 objects were segmented, and 2308 predecessors were assigned. In an extreme case, suppose an error occurs within each of the six lineages before any cell division (if there will be any), then the entire dataset could become completely useless, even though the error rate will be around 6/2314 for segmentation and around 6/2308 for tracking, which are extremely low.

In fact, on 'HaCaT-FUCCI' dataset, eDetect's automatic workflow reached an SEG of 0.9837 (Tables 3.2) and a TRA of 0.9870 (Tables 3.3) - the segmentation and tracking error rates were very low. However, the two more biologically relevant scores, $F_1(CT)$ and $F_1(CL)$ are 0.2482 (Tables 3.4) and 0.1081 (Tables 3.5), respectively. The cell lineages shown in Figure 3.11 will not be able to deliver valuable biological insights. Quantitative evaluation of the performances on the other three datasets also show that a low error rate (Table 3.3) does not guarantee a satisfactory reconstruction of cell lineages (Tables 3.4 and 3.5).

In summary, even a small portion of mistakes substantially compromise the validity of the final results. So it important to make the analysis free of errors. Since no automatic algorithm is error free, it seems inevitable to resort to manual correction after automatic analysis.

3.6.4 Combining automatic analysis and manual correction

Several tools have provided functions to manually correct results of automatic segmentation and tracking, such as CellProfiler [58,59], tTt/qTfy [87], CellTracker [136], LEVER [137], and NucliTrack [89].

Both CellTracker [136] and LEVER [137] require users to visually check every image in the video, though LEVER is able to automatically correct some errors to accelerate the process. With CellProfiler [58,59] users also need to look at the segmentation result of every image. And it does not have a regular window for image display. It processes one image per iteration, then it pauses and displays the newly processed image together with segmentation, allowing the user to visually check and correct one by one. This means the user has to stay with the tool throughout the entire workflow. CellProfiler Tracer [138] is able to visualize single cell traces but it is only designed for optimizing parameters for automatic algorithms.

tTt and qTfy [87] not only support editing segmentation and tracking results, but also provide an automatic outlier detection function for extracted time-series data (similar to the format of Figure 3.2 A). However, automatic algorithms are not absolutely reliable, thus still cannot guarantee an error-free final result. qTfy also uses a heatmap to visualize time-series data (similar to Figure 3.2 B), but the display interface is not used for interactive quality control. NucliTrack ([89]) visualizes time-series data and supports track-editing by clicking on any suspicious part. However, one lineage is visualized at a time, limiting the efficiency of improvement it could provide.

In summary, none of these tools provide an efficient error detection scheme to reach accurate final results with high efficiency. Here, we report eDetect, an error detection and correction tool for live cell imaging data analysis [1]. eDetect is a comprehensive software tool that integrates cell segmentation, cell tracking, and efficient data curation. It borrows the ideas of annotating cell lineages with morphological features for visualization from NucliTrack [89] and visualizing annotated cell lineages with heatmaps from qTfy [87]. In addition, eDetect also provides the options of automatic outlier detection and visualizing synchrograms as qTfy [87] and CellProfiler Tracer [138] do, respectively. The main contribution of eDetect, however, is using a 2D embedding to visualize nuclear morphological features for fast segmentation error detection and a gating strategy for fast segmentation error correction. A combination of these approaches together with an easy-to-use manual correction toolbox provide a comprehensive solution allowing efficient acquisition of accurate single cell dynamics.

3.7 Error detection and correction

As is discussed in section 3.6, automatic methods are important for the efficiency of data analysis, but high data quality requires post-editing. eDetect complements automatic approaches with four GUIs (graphical user interfaces) supporting data visualization based error detection and interactive error correction. They are designed for achieving both high efficiency and high accuracy in the acquisition of final results.

3.7.1 Main interface

Overview

The main interface (Figure 3.12) is the starting point of any data analysis session. In the top-left-most corner, four buttons each link to creating a project, loading a project, changing parameters and changing settings. After initiating a session, automatic data analysis modules (second top-most row) are able to give a preliminary data analysis result (section 3.3).

In addition, the main interface supports augmented image display and manual error correction. Here the user can browse through and get hold of an impression of the time-lapse video as well as the preliminary segmentation results given by the automatic analysis. It is also possible to look up close at tracking results. If the user spots any segmentation or tracking error, there are tools that support manual editing.

Even without the three other modules (segmentation gating, cell pair gating and cell lineages display), eDetect would still be an intact tool with automatic workflow and post-editing functionality, but the error detection and correction will be quite primitive and inefficient. In practice, users are supposed to carefully check the raw images and analysis results on main interface only when they detect errors with the help of one of the other three interfaces.

Augmented image display

The main interface is used for displaying images in the dataset, superimposed with object contours or masks if segmentation is finished. The image is displayed in the bottom right of the main interface (Figure 3.12). The navigation bar allows the user to browse through the dataset across scenes, frames and channels. The display control allows the user to adjust the curve (tonality) of the image to improve contrast or brightness.

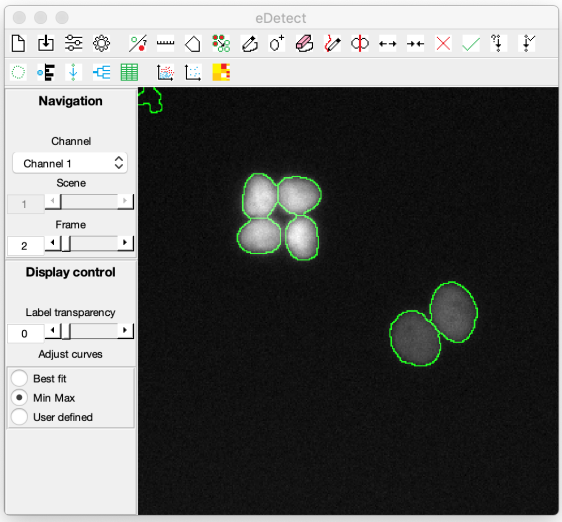


Figure 3.12: Main interface of eDetect. Top row: buttons for initiating a project and tools for image display and manual correction. Bottom-left: navigation and display control. Bottom-right: image display.

Manual correction toolbox

Another important function of the main interface is to allow users to manually correct segmentation and tracking results. It is possible to remove an object, draw and create an object, delete (mark as erroneous) an object, recover (mark as not erroneous) an object, draw a line to divide an object, make an object split into pieces, and merge several objects. It is also possible to check and edit the predecessor of any object, therefore allowing manual correction of tracking errors.

3.7.2 Segmentation gating

Motivation and ideas

The purpose of creating segmentation gating was to display all the detected objects from all scenes and all frames on one panel, enabling the user to see all of them at once and spot segmentation errors easily regardless of which image they are in. This will save the users from laboriously navigating on the main interface through frames and scenes of a time-lapse microscopy dataset, looking for errors.

Now we face two questions: (1) where to put each object and (2) what each object looks like. The purpose of collecting all the objects together is to make it easier to spot errors, so either the location or the appearance of an object should be able to imply how likely the object segmentation is to be incorrect. One very straightforward option is to display each segmented/cropped object and/or its mask directly on the GUI. Obviously the users will be able to see each object. However, when the total number of objects are large, either they do not fit or they degenerate to points with the decrease of resolution. Once given up sticking objects directly on the GUI we will have to come up with a way to transform the objects to a new representation and visualize this new representation. The new representation could be for example a vector of numbers for each object.

The strategy we adopted was to represent the objects with vectors of continuous values, embed the vectors into the 2D Cartesian coordinate plane with dimensionality reduction, and visualize them with a scatter plot. We could assign different markers and colors to the points in order to discriminate between different subsets (as will be explained later). Most importantly, using the points' location to carry information about the objects will possibly group similar objects together on the Cartesian coordinate plane, creating the potential of batch error correction - selecting a group of objects and correct them with one click.

Feature extraction

eDetect allows users to extract five categories of cellular features from object contours and nuclear marker images, i.e. shape (default), intensity, Haralick features [139], Zernike features [140] and additional features (default). The calculations of intensity, Haralick and Zernike features are implemented using CellProfiler source codes [58]. Shape features include 'Area', 'Eccentricity', 'Orientation', 'Solidity', 'Perimeter', and 'MeanIntensity' (see `https://www.mathworks.com/help/images/ref/regionprops.html`). 'Radial distribution' describes how intensity varies as a function of distance from the nearest point in the border of the object.

Data visualization

Here I will explain how the features of segmented objects from all scenes and frames become the points in one scatter plot. In short, this is done in two steps: feature mapping and 2D embedding.

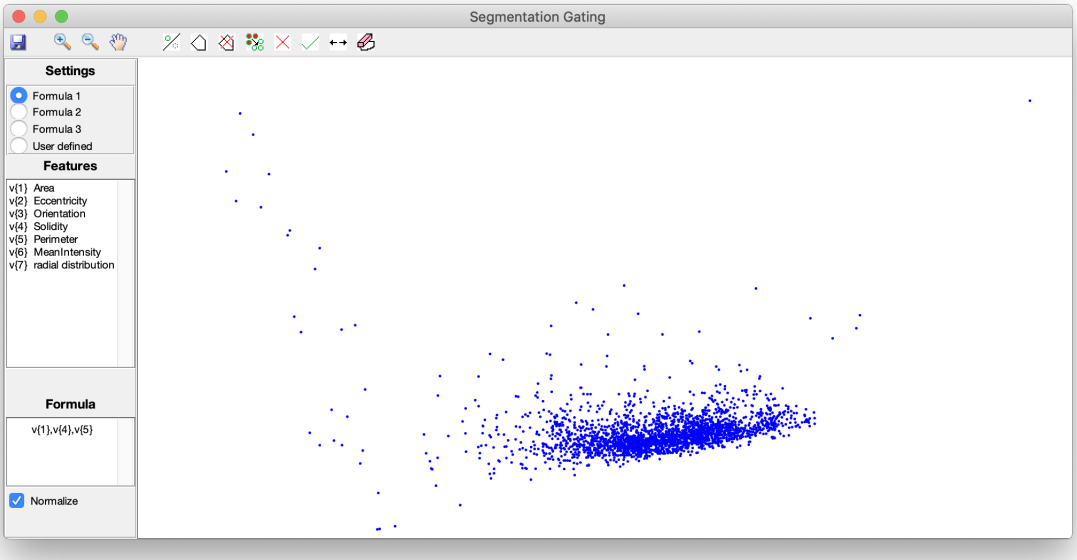


Figure 3.13: Segmentation gating. Left: settings, features and formula. Each 'setting' is a predefined 'formula'. The 'formula' consists of a few user-customized functions of 'features'. Right: scatter plot. Each point represent an object. x-axis: first principal component. y-axis: second principal component.

This feature extraction yields a variable matrix V of N_{obj} rows and N_{var} columns:

$$V = \begin{bmatrix} v_{1,1} & \cdots & v_{1,N_{var}} \\ \vdots & \ddots & \vdots \\ v_{N_{obj},1} & \cdots & v_{N_{obj},N_{var}} \end{bmatrix} = \begin{bmatrix} \vec{v_1} & \cdots & \vec{v_{N_{var}}} \end{bmatrix}, \quad \vec{v_1}, \vec{v_2}, ..., \vec{v_{N_{var}}} \in R^{N_{obj}}, \tag{3.10}$$

in which N_{obj} is the total number of objects in the entire dataset and N_{var} is the number of features that the user decides to extract. For example, the default features are 'Area', 'Eccentricity', 'Orientation', 'Solidity', 'Perimeter', 'MeanIntensity' and 'radial distribution'. So in this case $N_{var} = 7$. The columns of V are $\vec{v_1}, \vec{v_2}, ..., \vec{v_{N_{var}}}$ (Figure 3.13 Features).

In segmentation gating, not all calculated features are used for subsequent analysis (2D embedding), and the selected features are not necessarily directly used. Instead, users are allowed to select a subset from features listed in 'Features', or even customize a vector of functions of the variables (features) (Figure 3.13).

In this step, the feature matrix V became a function matrix F of N_{obj} rows and N_{fcn} columns:

$$F = \begin{bmatrix} f_{1,1} & \cdots & f_{1,N_{fcn}} \\ \vdots & \ddots & \vdots \\ f_{N_{obj},1} & \cdots & f_{N_{obj},N_{fcn}} \end{bmatrix} = \begin{bmatrix} \vec{f_1} & \cdots & \vec{f_{N_{fcn}}} \end{bmatrix}, \quad \vec{f_1}, \vec{f_2}, ..., \vec{f_{N_{fcn}}} \in R^{N_{obj}}, \tag{3.11}$$

in which N_{fcn} is the number of user customized functions. Each $\vec{f}$ is a function of all $\vec{v}$s: $\vec{f_i} = g_i(\vec{v_1}, \vec{v_2}, ..., \vec{v_{N_{var}}})$. For example, in Figure 3.13, $\vec{f_1} = \vec{v_1}$, $\vec{f_2} = \vec{v_4}$ and $\vec{f_3} = \vec{v_5}$. The functions could also be as complex as, e.g.: $\vec{v_1} \odot (\vec{v_2} - \vec{v_3} \textcircled{e} \vec{v_4} \oslash (\vec{v_5} + \vec{v_6}))$, in which $\odot$, $\oslash$, and $\textcircled{e}$ mean element-wise multiplication, division, and exponentiation, respectively. But usually simple functions are already sufficient and complex ones are difficult to interpret.

In the end the matrix F is transformed into a principal components matrix PC of N_{obj} rows and N_{fcn} columns:

$$[PC] = \begin{bmatrix} pc_{1,1} & \cdots & pc_{1,N_{fcn}} \\ \vdots & \ddots & \vdots \\ pc_{N_{obj},1} & \cdots & pc_{N_{obj},N_{fcn}} \end{bmatrix} = \begin{bmatrix} p\vec{c}_1 & \cdots & p\vec{c}_{N_{fcn}} \end{bmatrix}, \quad p\vec{c}_1, p\vec{c}_2, ..., p\vec{c}_{N_{fcn}} \in R^{N_{obj}}, \tag{3.12}$$

using PCA (principal component analysis) [141]. PCA converts a data table, in which the observations are described by a set of possibly correlated variables, into a new one, where the new variables (i.e.: principal components) are uncorrelated [17]. From the first PC (principal component) to the last one, each PC maximizes the data variance under the constraint that it is orthogonal to all the preceding ones.

In segmentation gating, $p\vec{c}_1$ and $p\vec{c}_2$ are used as the x- and y-axis variables in the scatter plots (Figure 3.13 scatter plot).

Error detection and correction

Here I am going to explain how to use segmentation gating for efficient segmentation error detection and correction.

To start the process the user should randomly click on dots and check them on the main interface. When a dot is clicked (when a dot is the closest to the clicked coordinate), the image display on the main interface automatically navigates to the image containing the object that this dot represents, and this object is highlighted. For example, when the dots marked by the red squares are clicked (Figure 3.14 A), the corresponding objects in Figure 3.14 B to F are displayed and highlighted, respectively. By checking the highlighted object on the main interface, the users will be able to tell whether the clicked object is a segmentation error. If there is an error the tools on the main interface allow the users to edit the results.

If the formula is properly customized, similar types of segmented objects may be positioned in groups. For example, in Figure 3.14, the points in the left of the scatter plot represent false positive objects (Figure 3.14 B and C), the points in the middle of the scatter plot mostly represent nuclei (Figure 3.14 D), and the points in the right of the scatter plot represent under-segmented objects (clusters/clumps of nuclei) (Figure 3.14 E and F).

This property - that similar types of objects are positioned in groups - makes efficient error detection and correction possible. After clicking on one dot and checking the object on the main interface, the users might be able to infer the type of objects that its close neighbors represent. After a few clicks the users could have a rough guess of what sorts of objects each region represent. Then they may need many more clicks to refine the boundary.

When users know the locations of the group of false positives (Figure 3.14 A left) and the groups of under-segmented objects (Figure 3.14 A right), they can select the target group and remove or split all the selected objects with one click. For example, in Figure 3.15 A the group of false positive objects are selected with a polygon. After 'Remove objects selected' is clicked (Figure 3.15 A), these dots are not in the scatter plot any more (Figure 3.15 B). In fact the objects that these dots represented are removed. Similarly, we can select the group of under-segmentations and click 'Split objects selected'. But this may not work all the time since some clustered nuclei are not automatically separable. If so we could click on each of them, locate them on the main interface, and divide them by manually drawing lines. In the end, the dots representing the under-segmentations will also disappear (Figure 3.15 C).

Note that using segmentation gating for error detection and correction may make mistakes, meaning that some errors may be overlooked, and some correct segmentations may be mistakenly removed or split, too. The rest of the errors will be spotted with the help of the subsequent error detection and correction interfaces.

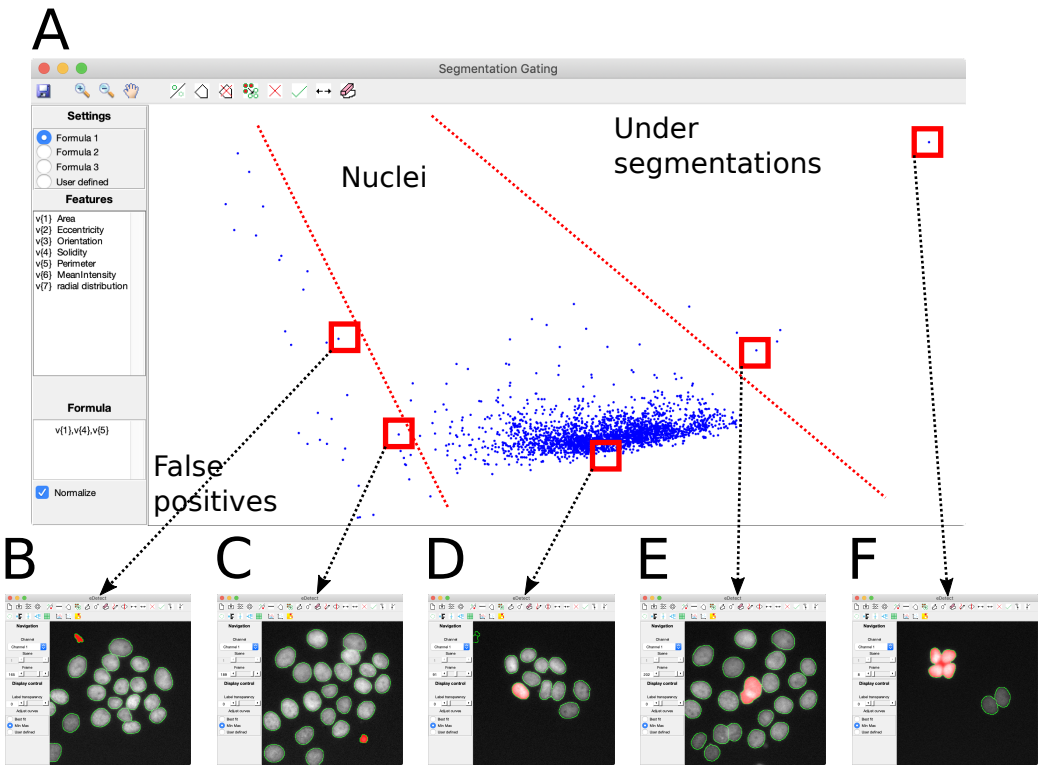


Figure 3.14: Segmentation gating enables efficient segmentation error detection. (A) Segmentation gating interface. Different groups of objects are divided by red dotted lines to make the groupings clear to see. Representative dots are marked with red squares and the objects they represent are shown in (B)-(F). (B)-(C) Examples of false positive objects. (D) An example of nuclei. (E)-(F) Examples of under-segmented objects.

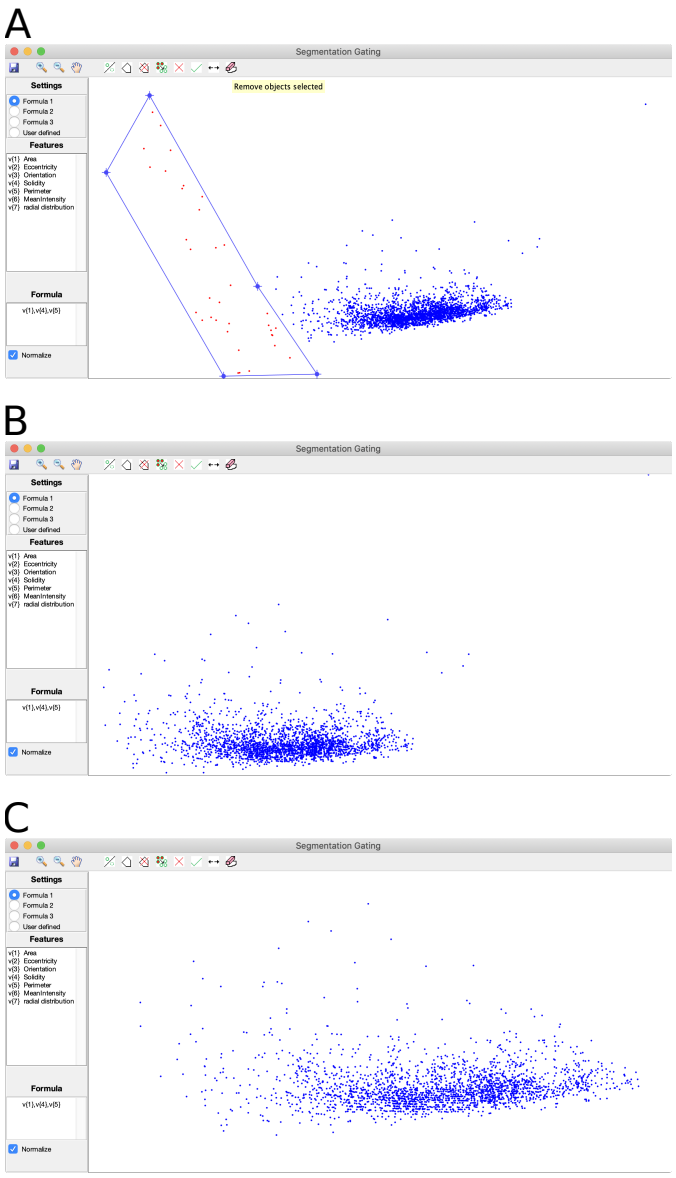


Figure 3.15: Segmentation gating enables efficient segmentation error correction. Panels (A) to (C) demonstrate a typical error correction process. (A) Segmentation errors of a certain type (false positives) are selected using the polygon selection tool. (B) After 'Remove object selected' is clicked, the selected dots disappeared. (C) After clicking the dots representing the group of under-segmented objects, and correcting them on the main interface one by one, those dots are also gone and the final segmentation panel looks like.

3.7.3 Cell pair gating

Cell pair gating helps users spot tracking errors the way they spot segmentation errors in segmentation gating. The overall design of cell pair gating is similar to that of segmentation gating. The appearances of 'settings', 'features' and 'formula' are the same (Figure 3.16).

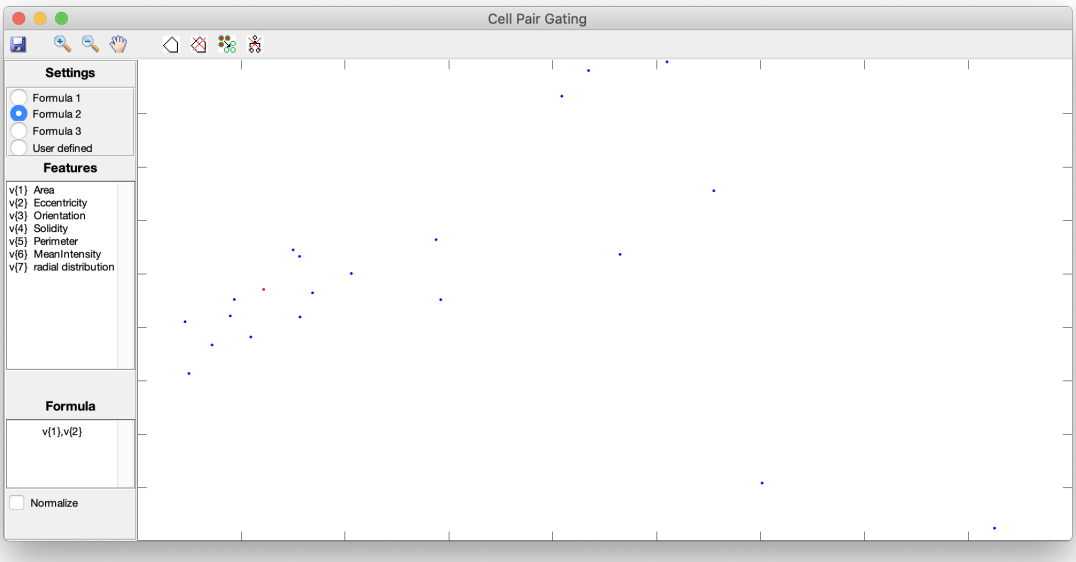


Figure 3.16: Cell pair gating. Left: settings, features and formula. Each 'setting' is a predefined 'formula'. The 'formula' consists of a few user-customized functions of 'features'. Right: scatter plot. Each point represent two cells that have the same predecessor. x-axis: first PC. y-axis: second PC.

In cell pair gating each dot in the scatter plot represents a pair of objects that share the same predecessor (Figure 3.16). Cell pairs certainly cannot reflect all tracking results. This data visualization is expected to expose a large proportion of tracking errors, since many tracking errors result in fake cell divisions.

Another difference is that, apart from the functions specified in the 'Formula', another three functions will be among the columns of the matrix (Equation 3.11) subject to PCA dimensionality reduction. These three functions describe the relative orientations and locations of the two objects within a cell pair. Specifically, let l_1 and l_2 be the major axes of the two objects, and let l_3 be the line that connects the centroids of the two objects. The three functions are defined as $\angle l_1, l_2$, $\angle l_2, l_3$ and $\angle l_3, l_1$. Here we use $\angle p, q$ to denote the sharp angle between lines p and q.

The usage of cell pair gating for tracking error detection and correction is also similar to using segmentation gating. By clicking on the scatter plot and checking the cell pair on the main interface, users see which part of the scatter plot has dots representing real cell divisions (Figure 3.17 A left, B) and which part has dots representing fake cell divisions (Figure 3.17 A right, C-E).

They may then dissociate them (let the two objects (sharing one predecessor) lose their predecessor) with polygon selection and one click, or check and correct each of them individually. Batch correction can largely reduce false positive CTs and CLs, but can also introduce new false negative CTs and CLs. Checking and correcting them one by one is slower but guarantees the quality of data.

Note that, after both segmentation gating and cell pair gating, there may still be undiscovered errors. These will most likely be spotted by the user in the next quality control step, the cell lineages display.

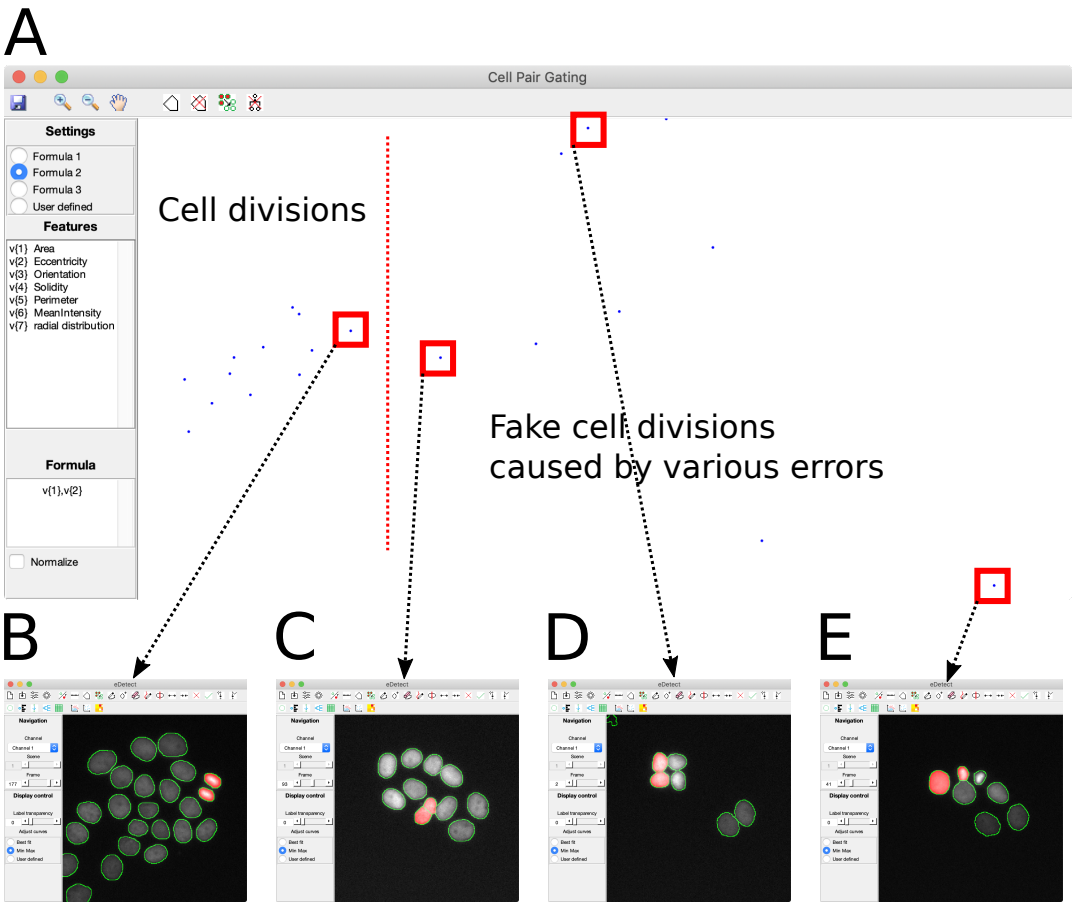


Figure 3.17: Cell pair gating enables efficient error detection. (A) Cell pair gating interface. Different groups of objects are divided by red dotted lines to make the groupings clear to see. Representative dots are marked with red squares and the objects they represent are shown in (B)-(E). (B) An example of real cell divisions. (C)-(E) Examples of fake cell divisions.

3.7.4 Cell lineages display

Overview

Cell lineages display is the last quality control step. It is supposed to expose all the segmentation and tracking errors that have not been discovered by the preceding quality control modules: segmentation gating and cell pair gating. After error detection and correction on cell lineages display, it is expected that the cell lineages and subsequently the final output are correct.

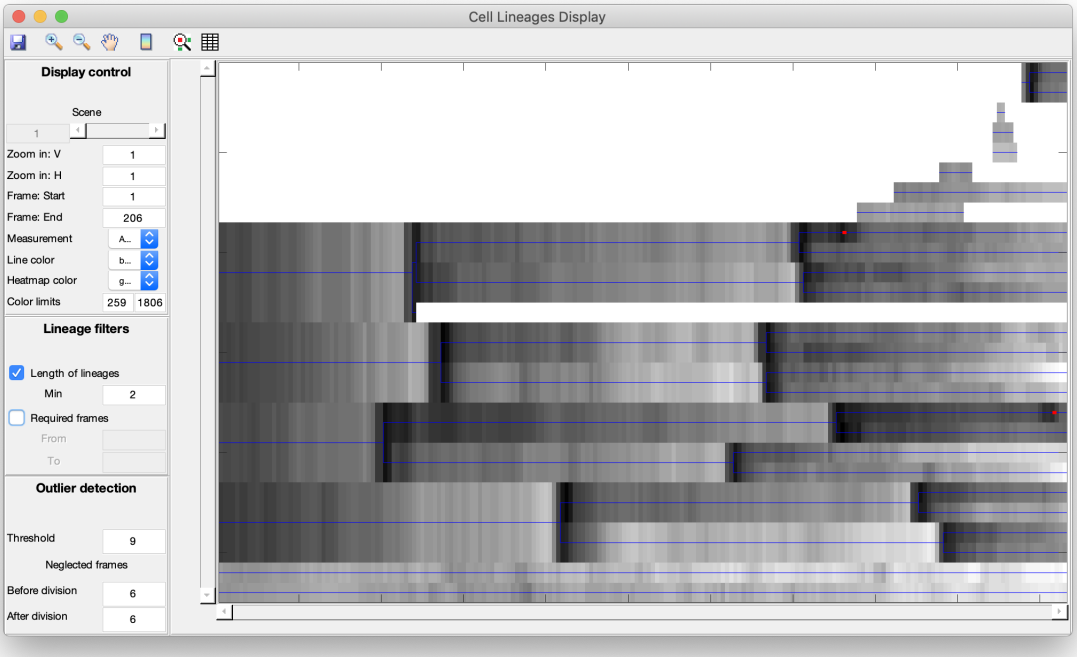


Figure 3.18: Cell lineages display. Left: display control, lineage filters and outlier detection. Right: heatmap.

Displaying cell lineages with a heatmap

In cell lineages display, a heat map (Figure 3.18) visualizes a 'feature matrix', which is calculated using a 'cell lineage matrix'. These two matrices are similar to the final cell lineages and time-series data (Figure 3.4 A and B). However, there are three differences: (1) the final results are free of errors but the cell lineages display is not; (2) in the final results usually a molecular reporter intensity is used to annotate lineages but cell lineages display usually uses a user-selected feature; and (3) the final results usually only contain CLs but cell lineages display also include incomplete ones.

In the 'cell lineage matrix', each column represents a frame, each row represents a cell lineage, and each entry is a cell ID (see 3.3.1). Each cell lineage corresponds to a unique 'terminal-state' object. Here a 'terminal-state' object is either an object in the last frame of the video, or an object that does not have a successor in the subsequent frame. Each 'cell lineage' is defined as a terminal-state object together with its entire history, including all the snapshots of itself and its mother cell and so on, if they existed. Obviously, if a cell divided, the object IDs representing the mother cell could be part of the history of multiple terminal-state objects and therefore part of multiple lineages. In this case, multiple lineages will partially share the same sequence of object IDs, just like in Figure 3.4 A. Because the rows are sorted in dictionary order, in which the 'words' are made up of 'letters' of object IDs, these lineages will be displayed close to each other vertically. The later two different lineages diverged, the closer they are arranged.

The 'feature matrix' is calculated by substituting each entry (object ID) of the 'cell lineage matrix' with the object's feature value (e.g. 'object area'). The higher this value is, the brighter the corresponding entry of the heat map is (Figure 3.18).

Cell lineages are also highlighted with blue lines. A cell division is indicated by a 'bifurcation' unless one of the daughter cells is filtered out from display, in which case a vertical line segment is used to mark the division event. Automatically detected outliers are highlighted in red horizontal line segments (Figure 3.18).

As has been mentioned, not all lineages are complete. For example, the lineages in the upper part of the heatmap have empty (white colored) entries (Figure 3.18). The lineages with white entries to the left did not start from the first frame, and are named 'late-started' lineages. The tracking algorithm did not find a predecessor for their starting objects. The lineages with white entries to the right did not extend until the last frame, and are named 'early-ended' lineages. The tracking algorithm did not find a successor for their ending objects.

Error detection with heatmap

Bifurcations, outliers, 'late-started' lineages and 'early-ended' lineages are useful starting points for error spotting. Here we give two examples.

In the first example, we discovered an error at two temporally close bifurcations, or cell divisions (Figure 3.19 A). The region of interest is zoomed in for clearer visualization (Figure 3.19 B). We are suspicious of the two cell divisions because we know the time interval between two frames is 15 minutes, and cells usually do not divide twice within 15 minutes. After clicking on the entries, we see the objects they represent on the main interface (Figure 3.19 C to G). It seems that the automatic tracking algorithm thinks both E and G are successors of D, but in fact, E is the successor of D, while G is the successor of F. This error was not spotted with the help of cell pair gating. In fact, the appearances of E and G did not change much from D and F, so they still look like a pair of newborn sister cells. This makes the dot representing them positioned close to the dots representing real newborn sister cells in cell pair gating. Correcting this tracking error requires a few mouse clicks on the main interface.

In the second example, we pay attention to automatically detected outliers (Figure 3.20 A). The outlier on the top-left is zoomed in and displayed in Figure 3.20 B, and the outlier on the bottom-right is zoomed in and displayed in Figure 3.20 E. In Figure 3.20 B, the red horizontal bar marks two horizontally consecutive entries, each representing the objects in Figure 3.20 C and D. The highlighted object in C is the predecessor of the highlighted object in D. Outlier indicates that abrupt change in the value of the displayed feature, i.e. object area, is detected, between the objects in C and D. Looking at the objects in C and D, we could see that there is no segmentation error or tracking error, but the segmentation accuracy is limited by the fact that nuclei are overlapping with each other. There is no good way to recover all the regions of each nucleus perfectly. The example shown in Figure 3.20 E to G presents a similar situation.

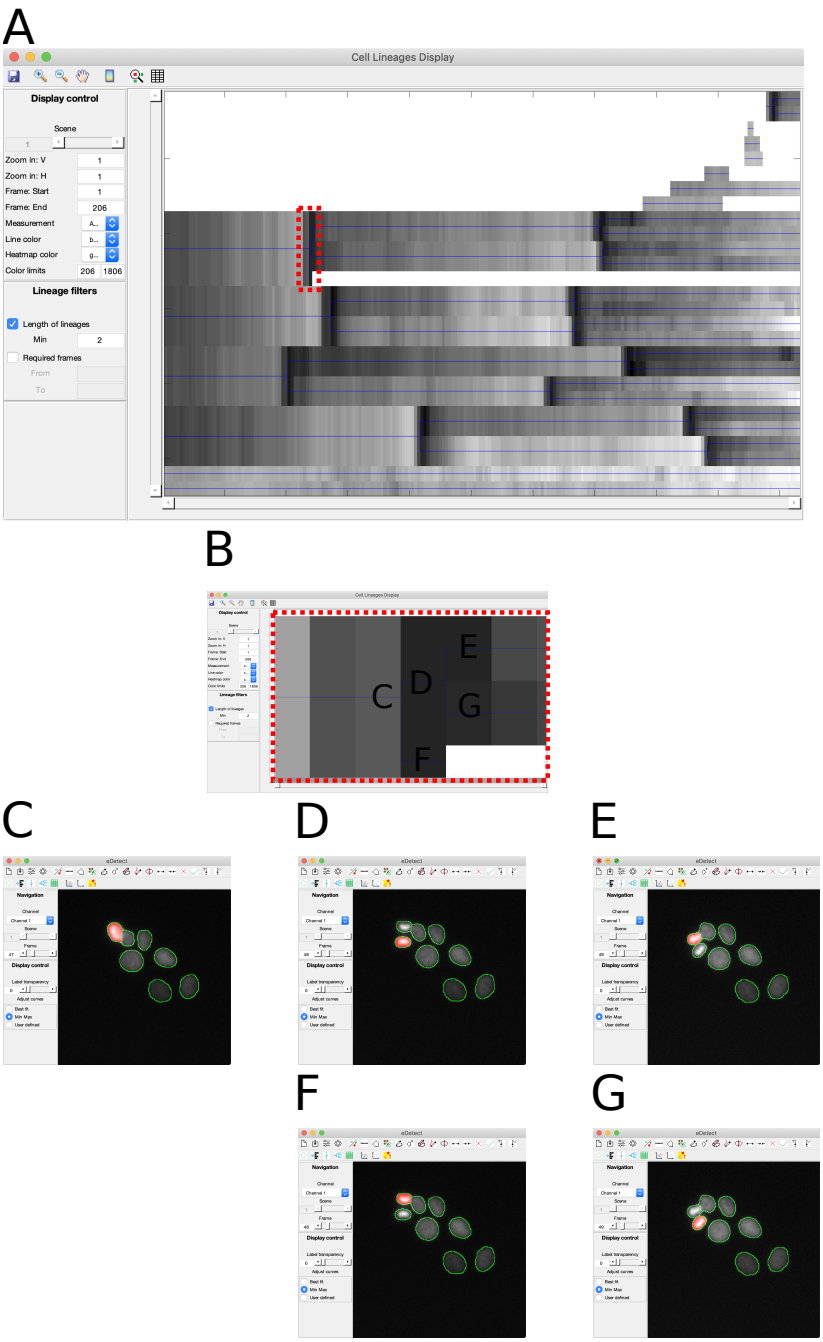


Figure 3.19: Error detection using cell lineages display (part 1). Two consecutive cell divisions easily catch users' attentions (A), which is zoomed in and displayed in (B). One cell (C) divided into (D) and (F), and immediately afterwards, (D) divided into (E) and (G). Clicking at the heatmap entries leads to main interface, which displays and highlights the objects that the clicked entries represent. Both (E) and (G) are identified as successors of (D).

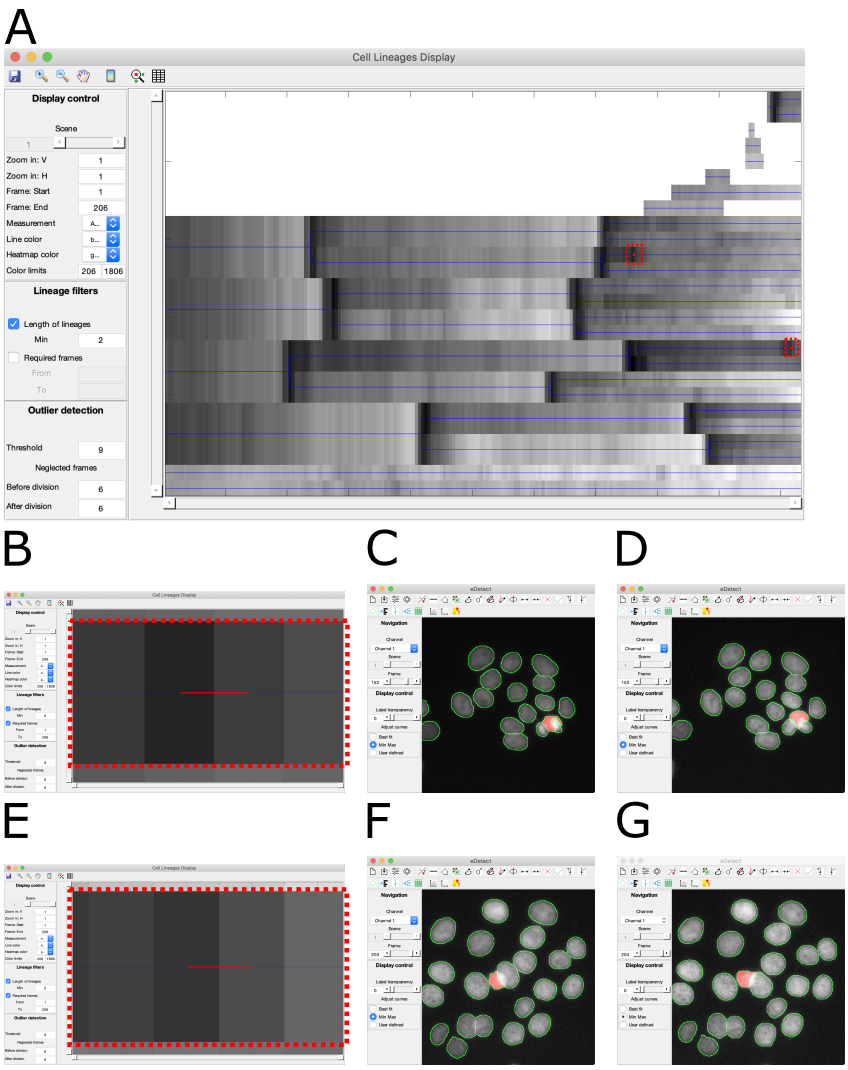


Figure 3.20: Error detection using cell lineages display (part 2). Two outliers are detected by automatic outlier detection (A). The one in top-left is zoomed in and displayed in (B), and the one in bottom-right is zoomed in and displayed in (E). The two marked entries in (B) represent highlighted objects in (C) and (D). The two marked entries in (E) represent highlighted objects in (F) and (G).

Synchrograms

Apart from the heat map, the cell lineages display additionally provides an option of opening synchrograms, which is a mosaic of the sequence of cropped image patches of all objects in a selected (right-clicked) cell lineage (3.21). Clicking on any of the patches also guides the user to the main interface, where the image containing the clicked object will be displayed and the clicked object will be highlighted. A similar function has been implemented in CellProfiler Tracer [138].

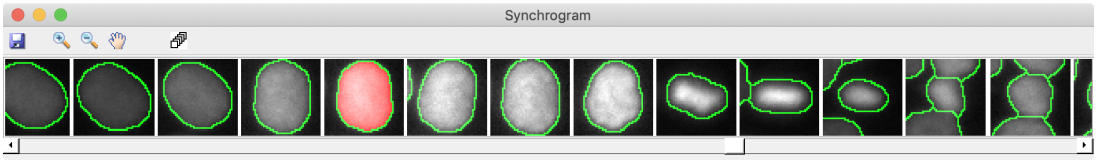


Figure 3.21: Synchrogram in the cell lineage display is a mosaic of the sequence of cropped image patches of all objects in a cell lineage. The highlighted object is the one right-clicked on the heatmap. These image patches are clickable. After clicking on an image patch, the corresponding frame will be displayed on the main interface, and the clicked object will be highlighted.

3.7.5 Workflow

In this section, I have presented the four GUIs of eDetect (Figure 3.22 grey trapezoids) based on the automatic computational workflow (Figure 3.5 C, Figure 3.22 rectangles and ellipses). The four interlinked GUIs (arrows between grey trapezoids) visualize raw and intermediate data (arrows from green and blue ellipses to grey trapezoids) and support interactive edition of the visualized data (arrows from grey trapezoids to blue ellipses).

The main interface allows users to directly look at the raw images and analysis results, which is easily editable with the help of a wide range of tools. Furthermore, segmentation gating visualizes all the segmented objects in one scatter plot, making it possible to detect and correct many segmentation errors with a few clicks, without the need to look through the entire video. The subsequent cell pair gating visualizes all the pairs of objects sharing the same predecessor, some of which may correspond to segmentation or tracking errors. In the end, the cell lineages display presents reconstructed cell lineage trees in a format that is very close to the final output, improving the quality of the entire analysis.

In summary, eDetect adopts a multistep strategy for the quality control of automatic analysis of live cell imaging. First, segmentation gating helps users detect as many errors as possible in a fast way, but probably leaving some errors unresolved. Then in cell pair gating users will spot most of the overlooked errors, and correct them one by one, leaving an ideally very low number of still overlooked errors. In the end, the cell lineages display exposes all the still existing errors, and users expect to acquire absolutely correct results after this step. This combination of the three different manual correction modules complemented by the main interface is designed for achieving very high accuracy within as short a time as possible.

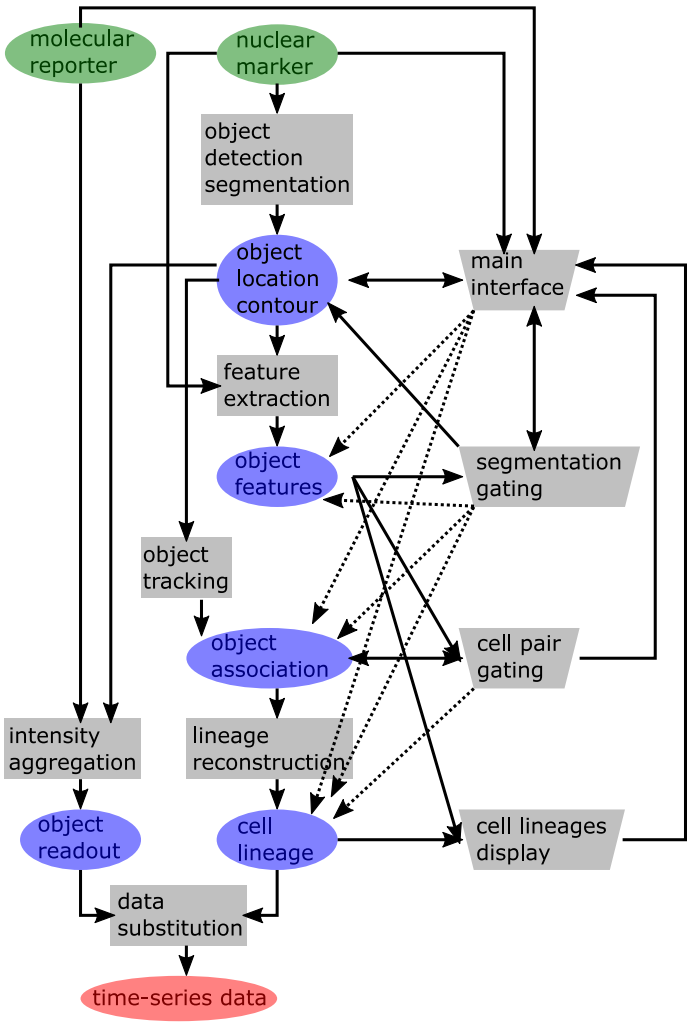


Figure 3.22: Workflow of the eDetect including computational pipeline and manual correction modules. Green ellipses: input data sources (images). Grey rectangles: automatic analysis modules. Grey trapezoids: manual analysis modules. Blue ellipses: intermediate results. Red ellipses: final output. Solid arrows from data to automatic module means the module takes the data as input. Solid arrows from automatic module to data means the module generates the data as output. Solid arrows from data to a manual correction module means the data is visualized in this module. Solid arrows from a manual module to data means the data are able to be corrected in this module. Dashed arrows pointing from a manual module to an intermediate result mean that the when manual correction is conducted on the manual module, some other data will be corrected, and this data will be also be updated immediately. Solid arrows from a manual module to another manual module means clicking on the former module links to the latter.

3.8 Manual correction improves performance

Error detection and correction modules (section 3.7) are implemented in eDetect to tackle the challenges (section 3.6) that live cell imaging data analysis has been facing. In this section, I will present to what extent eDetect can improve the performances of automatic analysis.

3.8.1 Performance on live cell imaging datasets

eDetect is used for manually correcting the automatic analysis results on four live cell imaging datasets, namely 'HaCaT-FUCCI', and 3 CTC benchmarking datasets 'Fluo-N2DH-GOWT1', 'Fluo-N2DL-HeLa' and 'Fluo-N2DH-SIM+' (subsection 3.2.1). The SEG (Table 3.7), TRA (Table 3.8), $F_1(CT)$ (Table 3.9) and $F_1(CL)$ (Table 3.10) scores are measured again and compared to those of automatic analysis. The manual correction time consumption was also recorded (Table 3.11).

In Tables 3.7, 3.8, 3.9 and 3.10, 'CellProfiler' and 'CellProfiler+' mean the results of CellProfiler automatic analysis before and after eDetect manual correction. 'eDetect' and 'eDetect+' mean the results of eDetect automatic analysis before and after eDetect manual correction. 'CTC 1st', 'CTC 2nd' and 'CTC 3rd' mean the highest, the second highest and the third highest scores from all CTC candidates, respectively.

Comparing 'CellProfiler' with 'CellProfiler+' and then 'eDetect' with 'eDetect+', we see that after manual correction, the SEG and TRA scores of both CellProfiler and eDetect were consistently improved (Tables 3.7 and 3.8). After manual correction, the performance of eDetect on SEG and TRA are comparable to the highest three scores from CTC candidates (Tables 3.7 and 3.8).

Table 3.7: Benchmarking segmentation before and after correction with SEG

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.6826	0.8300	0.7256	0.7794	0.8287	0.2584	0.6841	0.8283
CellProfiler+	0.8380	0.8647	0.7775	0.8298	0.8366	0.6005	0.7912	0.8630
eDetect	0.7449	0.9377	0.8065	0.8473	0.8364	0.5373	0.7850	0.9837
eDetect+	0.8798	0.9381	0.8230	0.8615	0.8473	0.6160	0.8276	0.9931
CTC 1st	0.7724	0.9413	0.8270	0.8469	0.8652	0.6577	0.8184	-
CTC 2nd	0.7262	0.9140	0.8251	0.8457	0.8640	0.6406	0.8026	-
CTC 3rd	0.7221	0.8980	0.7774	0.8140	0.8508	0.6298	0.7820	-

Table 3.8: Benchmarking tracking before and after correction with TRA

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.9470	0.9467	0.8982	0.9244	0.9727	0.3638	0.8421	0.9695
CellProfiler+	0.9941	0.9938	0.9368	0.9374	0.9870	0.9160	0.9609	0.9987
eDetect	0.9279	0.9465	0.9707	0.9682	0.9817	0.3249	0.8533	0.9870
eDetect+	0.9942	0.9866	0.9803	0.9763	0.9932	0.9180	0.9748	0.9988
CTC 1st	0.9654	0.9746	0.9858	0.9794	0.9926	0.9501	0.9747	-
CTC 2nd	0.9617	0.9578	0.9804	0.9665	0.9916	0.9212	0.9632	-
CTC 3rd	0.9597	0.9532	0.9762	0.9620	0.9876	0.8918	0.9551	-

Table 3.9: Benchmarking tracking before and after correction with $F_1(CT)$

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.0541	0.1881	0.0556	0.1753	0.3122	0.0057	0.1318	0.0252
CellProfiler+	0.8302	0.6168	0.6361	0.3226	0.7608	0.7103	0.6461	0.8293
eDetect	0.0237	0.0245	0.3208	0.1271	0.3596	0.0006	0.1427	0.2482
eDetect+	0.8302	0.6364	0.6469	0.1915	0.8172	0.7426	0.6441	0.9189
CTC 1st	0.7547	0.5984	0.4678	0.4106	0.7500	0.0878	0.5116	-
CTC 2nd	0.6667	0.5918	0.4517	0.3243	0.6878	0.0870	0.4682	-
CTC 3rd	0.5938	0.5652	0.3053	0.2397	0.5376	0.0695	0.3852	-

Table 3.10: Benchmarking tracking before and after correction with $F_1(CL)$

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler	0.7317	0.6316	0.4809	0.3973	0.4444	0.0000	0.4477	0.0541
CellProfiler+	0.9500	1.0000	0.8807	0.5859	0.9882	0.8842	0.8815	1.0000
eDetect	0.9048	0.9032	0.7164	0.6007	0.4524	0.0000	0.5963	0.1081
eDetect+	0.9500	1.0000	0.9286	0.8015	0.9882	0.8660	0.9224	1.0000
CTC 1st	0.9743	0.9655	0.7778	0.5908	0.9882	0.5684	0.8108	-
CTC 2nd	0.9500	0.9630	0.5258	0.4361	0.9762	0.1887	0.6733	-
CTC 3rd	0.9474	0.9286	0.3046	0.3149	0.9639	0.1818	0.6069	-

Table 3.11: Time consumption of manual correction

Dataset	Fluo-N2DH-GOWT1		Fluo-N2DL-HeLa		Fluo-N2DH-SIM+		Mean	HaCaT-FUCCI
Video	1	2	1	2	1	2	-	-
CellProfiler+	18min	20min	55min	1h12min	15min	4h11min	-	16.5 min
eDetect+	15min	20min	30min	1h19min	9min	3h44min	-	9min

More importantly, biologically relevant scores, $F_1(CT)$ and $F_1(CL)$, are improved to a larger extent after manual correction (Tables 3.9 and 3.10). The $F_1(CT)$ and $F_1(CL)$ scores of 'eDetect+' are almost consistently higher than 'CTC 1st'. There are only three exceptions. The first is the $F_1(CT)$ of the 2nd video of 'Fluo-N2DL-HeLa'. In this video, the correct segmentation of many nuclei were unclear to the corrector, which obviously limited the possible performance improvement brought by manual correction. The second is the $F_1(CL)$ of the 1st video of 'Fluo-N2DH-GOWT1'. The scores of both 'eDetect+' and 'CTC1' are close to each other and close to 1. The third is the 1st video of 'Fluo-N2DH-SIM+', with which 'eDetect+' achieved the same score as 'CTC1'.

Despite the performance improvements, the time consumption is acceptable (Table 3.11). All the manual correction times are shorter than 90 minutes except with the 2nd video of 'Fluo-N2DH-SIM+'. Poor automatic analysis results on this simulated video with artificial noise require substantial manual segmentation.

The improvement on scores $F_1(CT)$ and $F_1(CL)$ are most prominent with long term live cell imaging datasets 'HaCaT-FUCCI'. The SEG scores by CellProfiler and eDetect improved slightly from '0.8283' to '0.8630' and from '0.9837' to '0.9931', respectively (Table 3.7). The TRA scores by CellProfiler and eDetect improved slightly from '0.9695' to '0.9987' and from '0.9870' to '0.9988', respectively (Table 3.8). However, the $F_1(CT)$ scores by CellProfiler and eDetect improved from '0.0252' to '0.8293' and from '0.2482' to '0.9189', respectively (Table 3.9). The $F_1(CL)$ scores by CellProfiler and eDetect improved from close to 0 to 1 (Tables 3.10). The improvement of quality of time-series data of 'HaCaT-FUCCI' is also obvious when visualized with a heat map in cell lineages display (Figure 3.23). Notably, the results of 'CellProfiler' and 'eDetect' were improved in only 16.5 minutes and 9 minutes, respectively.

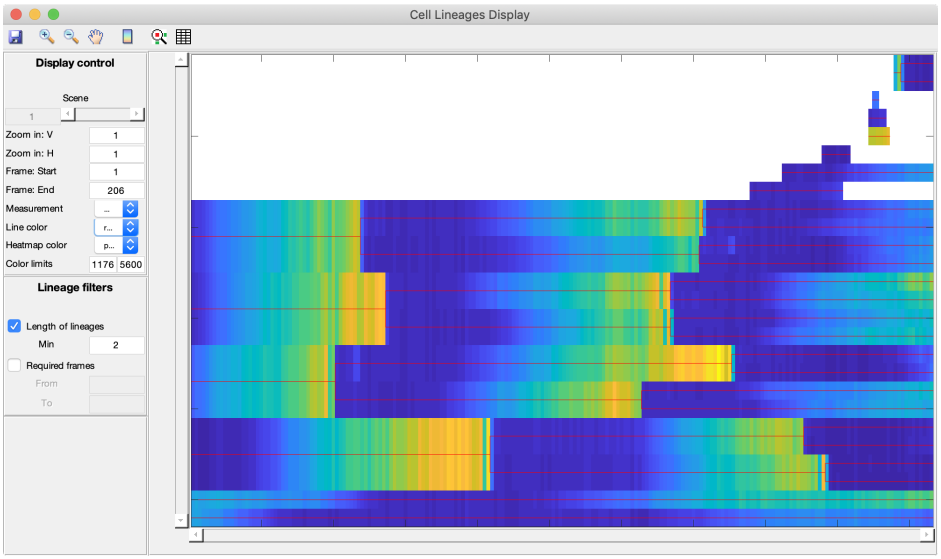


Figure 3.23: Time-series data acquired from eDetect automatic analysis and manual correction is displayed in heatmap. The original dataset is 'HaCaT-FUCCI'. The color of the heatmap entries encodes the time-series of mCherry-Geminin nuclei intensity. The structure of the heatmap demonstrates the cell lineages.

3.8.2 Performance on high-throughput screening dataset

The dataset 'BBBC039' is used for evaluating the improvement on F_1 score of segmentation before and after manual correction.

After manual correction both the results of eDetect automatic analysis and CellProfiler automatic analysis are largely improved (Figure 3.24). When the standard is not very strict (IoU threshold lower than 0.8), the performance of eDetect after 30 or 60 minutes of correction is comparable with that of 'unet4nuclei'. 'unet4nuclei' is a U-Net [4] package developed for nuclei segmentation [129] and it was trained on a training set of 100 images and validated with 50 images.

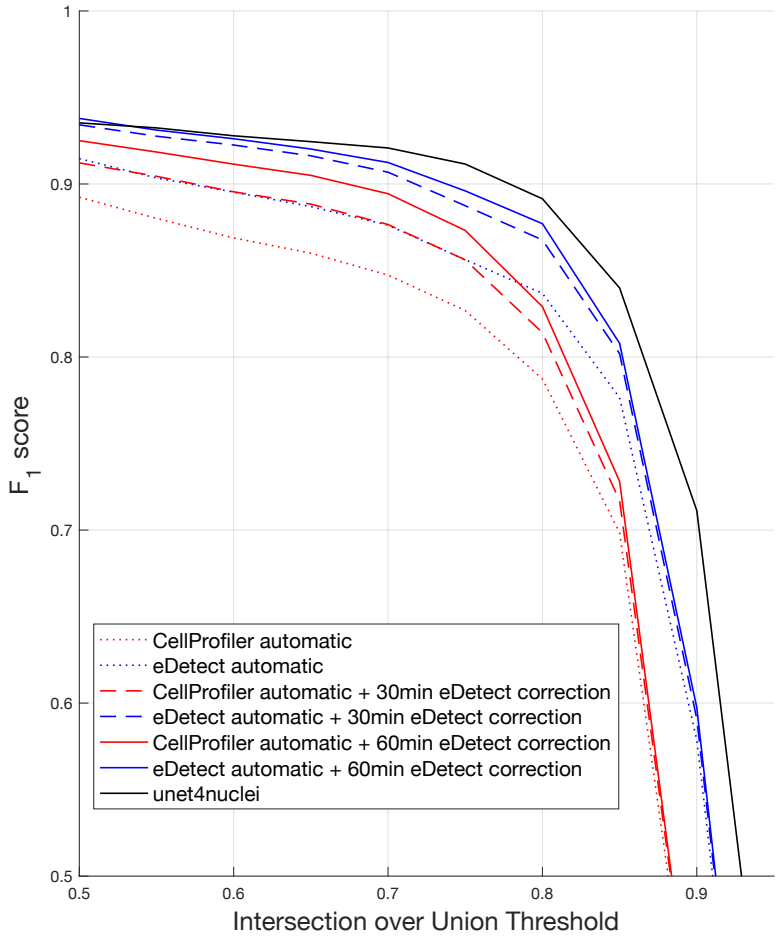


Figure 3.24: Evaluating the improvement of performance with manual correction based on F_1 scores of segmentation. x-axis: the minimal IoU (Jaccard index) to consider two segmented objects to be a match. y-axis: F_1 score of segmentation. Red dotted line: performance of CellProfiler automatic analysis. Blue dotted line: performance of eDetect automatic analysis. Red solid line: performance of CellProfiler automatic analysis and eDetect manual correction. Blue solid line: performance of eDetect automatic analysis and eDetect manual correction.

3.9 Methods

3.9.1 Segmentation

Adaptive thresholding is used to binarize greyscale images

A median filter (subsection 2.3.1) is applied on the raw image in order to remove possible salt-and-pepper noise, then the denoised image is smoothed using a Gaussian filter. Afterwards, minimum value is subtracted from the smoothed image.

The minimum-subtracted image is fed into six adaptive thresholding [8] functions (subsection 2.3.5) with different settings and parameters in parallel, producing six binary images. The six binary images are then combined into one binary image with an 'or' logical operator. This means a pixel in the output image is 'true' as long as at least one of the six images takes the value 'true' at this pixel. The combination of multiple adaptive thresholding trades specificity for sensitivity. This strategy is used because it is easier to correct false positives (with mouse clicks) than false negatives (with freehand drawing).

In the end, the combined binary image is subject to four morphological operations (subsection 2.3.6) as post-processing: 'binary image opening', 'binary image area opening', 'binary image closing' and 'binary image hole filling'. These operations are able to remove noise and make the object contours smoother.

Watershed transformation is used to separate clustering nuclei

The binary image from the previous step is inverted, resulting in a new binary image where background is filled with 1s and nuclei are marked with 0s. This new binary image is then used to calculate a distance map. In the distance map of a binary image, the pixel value is 0 where the original image is 1, and the pixel value is the shortest distance to 1 where the original image is 0. So in the distance map, background is filled with 0s, and each pixel value in nuclei is the shortest distance to background. Then an H-maxima transform is used to suppress those local maxima below a certain threshold [57]. This step prevents some objects from possessing too many local maxima, therefore decreasing the chance of over-segmentation.

This filtered distance map is then inverted, making each object into a 'pit' with possibly multiple local minima. Then the background is set to minus infinity. Then a watershed transform [142] separates 'pits' (objects) with multiple local minima into multiple objects. Finally, the background is set to 0, the image is binarized with a threshold of 0, and small objects are removed with an 'area opening' morphological operation.

Over-segmented objects are merged

Because single nuclei sometimes incorrectly split into smaller objects, some directly adjacent objects need to be merged. eDetect decides which pairs of adjacent objects to merge in a group of connected objects, based on the assumption that the shape of nuclei are close to ellipses. Here, two objects are connected if they are at two ends of a 'sequence' of objects, in which neighbouring ones are adjacent. A group of objects are connected if they are mutually connected (similar to pixel connectivity in subsection 2.3.7).

We transform the task into searching for the status with the lowest total area of minimum enclosing ellipses in a depth-first manner within the entire space of possible statuses. Here, for each object, its minimum enclosing ellipse is the smallest ellipse that is able to completely cover the object [143]. While for each connected group of objects, its total area of minimum enclosing ellipses is the sum of minimum enclosing ellipses of every object in the group.

In the space of possible statuses, it is possible to move from one status to another by merging exactly one pair of adjacent objects. We start from the original status where every object is on its own. In the end status, all objects in the group are merged into one single object. There could be many intermediate statuses, and many paths to travel from the original status to the end status. This procedure is similar to that of used by Lin *et al.* [144].

3.9.2 Tracking

In eDetect, object (nucleus) tracking is done with a constrained nearest-neighbor strategy. For each object (except those in the first frame), its distances to all the objects in the previous frame are calculated. The closest one in the previous frame is then determined to be the predecessor, unless this shortest distance exceeds a predefined threshold.

Sometimes the fields of view of neighboring frames are not precisely aligned, due to the instability of microscope systems. This problem is tackled using an optimization approach. For each frame, before calculating distances to the objects in the previous frame, a 'shift vector' is subtracted from the x-y coordinates of all objects of the current frame. Then there will be at least one 'shift vector' (searched within a square whose size is predefined) that makes the sum of all shortest distances reach the minimum. Then this 'shift vector' is finally applied, and the predecessors that correspond to this minimal sum are the final tracking result.

3.9.3 Measurements

The following measurements are calculated for each fluorescent molecular reporter channel: nuclei median intensities, nuclei mean intensities, cytoplasm median intensities, cytoplasm mean intensities, nuclear-cytoplasmic ratios of medians and nuclear-cytoplasmic ratios of means.

To accurately measure aggregated nuclear intensities, possible nuclear membrane pixels need to be excluded. The intensity aggregation region for nuclei are calculated by shrinking the nuclear segmentations by a user-defined distance. Similarly, to accurately measure aggregated cytoplasmic intensities, nuclear membrane, cell membrane and cell exterior pixels need to be excluded. At the same time, the measured cytoplasmic region needs to be large enough so that the measurement is robust. So the cytoplasmic regions are approximated with ring regions around nuclear segmentations. The inner and outer contours of these rings are both grown from nuclear segmentations but by different user-defined distances. To compensate for medium illumination, median or mean background intensity is subtracted from both nuclear and cytoplasmic intensities. The background region is derived by removing the foreground region from the whole image. To produce the foreground, nuclear segmentations are expanded by a relatively large distance, which is the maximum between two values: (1) user-defined maximum of object diameters, and (2) twice the distance by which nuclear segmentations are expanded while generating the outer contour of the ring regions around the nuclei.

Shrinking and expanding an object is defined as such: shrinking an object by n means iteratively removing a layer of pixels from contour interior n times; expanding an object by n means iteratively adding a layer of pixels to the contour exterior n times.

3.9.4 Outlier detection in cell lineages display

The cell lineages display supports automatic detection of outliers among all the differences in visualized measurement values between consecutive frames. Among all the differences, outliers are defined as values outside the range, which is centered on median and has a radius of a certain predefined number of standard deviations [145]. The outliers reflect abrupt changes in measurement values.

Chapter 4

Discussions

4.1 Summary

In this thesis, I described a software tool called 'eDetect' designed to tackle a challenge that threatens to impair almost all automatic approaches - imperfect accuracy. Instead of seeking a perfectly fine-tuned expert system that never makes mistakes, we adopted a more tolerant and robust design that allows the automatic analysis to make some mistakes, which will be resolved subsequently in a semi-automatic manner. The main contribution of eDetect is the rationalization and improvement on efficiency of manual error detection and correction.

In **segmentation gating**, the distances between two dots reflect the distances between segmented objects in the feature space. Therefore segmented objects with similar appearances will likely be positioned in each others' vicinity. Thus, when the user finds a dot representing a segmentation error, there is a good chance that the nearby dots also represent the same type of erroneous objects. After some mouse clicks the user should be able to roughly determine the border of the territory occupied by segmented objects affected by this type of error. With the gating tool, the user is free to draw a polygon to select dots, and choose to remove them or separate them into smaller parts.

Cell pair gating inherits the overall design of segmentation gating. But each dot on its scatter plot now represents a pair of objects that are assigned the same predecessor. Ideally each of them should be a pair of newborn sister cells. In reality segmentation and tracking errors also cause two cells that are not newborn sister cells to have the same predecessor. Cell pair gating is able to unveil these

segmentation and tracking errors. Similar to segmentation gating, similar cell pairs are likely closely distributed. This again makes it easier to discover more similar errors once the first is spotted.

Users are expected to correct a large portion of segmentation errors efficiently in the first interactive module (segmentation gating). In the second module (cell pair gating), the user is supposed to discover and correct most of the tracking errors, as well as most of the segmentation errors overlooked in segmentation gating. In the last step, **cell lineages display**, the data are visualized in a more structured and informative way: a user-selected feature (e.g. object area) will be used to annotate cell lineage trees. This information-rich data visualization makes it possible for users to visually examine the correctness of cell lineages based on two assumptions from biological domain knowledge: (1) cell cycle lengths vary only in an acceptable range, and (2) a cell's appearance usually only change gradually within interphase. Any violations of these principles, such as very frequent divisions or abrupt changes in lineage annotation colors, are highly suspicious and require immediate close-up examination. Fortunately this strategy is able to expose all errors most of the time. The sensitivity of abrupt change detection is determined by a user's visual checking, which means this procedure is as sensitive as the user want it to be. But to assist users, we still added an 'outlier detection' function to automatically detect abrupt changes in cell lineage branches. We also provided the option of opening synchrograms, which displays a mosaic of one cell's snapshots in all the frames. These additional features give users more choices and increase the chance of spotting errors. The manual correction workflow made up of these GUIs should be able to deliver accurate cell lineage trees. The resulting time-series data will be ready for downstream analyses.

Leveraging these interactive modules, segmentation and tracking performances were largely improved, especially on the more biologically relevant metrics: F_1 scores of segmentation, CT and CL (Tables 3.2, 3.3, 3.4, 3.5, 3.7, 3.8, 3.9, 3.10 and Figures 3.10, 3.24). As a result, the performances of eDetect with manual correction on CTC datasets are overall comparable to the best performances from CTC candidates on SEG (Tables 3.7) and TRA (Tables 3.8). More importantly it surpasses the best performances from CTC candidates on $F_1(CT)$ (Table 3.9) and $F_1(CL)$ (Table 3.10) in most test cases. The performances of eDetect with manual correction on high-throughput screening datasets is comparable to U-Net if high segmentation accuracy is not required (Figure 3.24).

In summary, eDetect offers a heuristic strategy to progressively improve the accuracy to a desired level. We expect it to be a preferred tool for projects that involve large datasets and require high accuracies.

4.2 Limitations

eDetect is aimed at improving manually corrected results, instead of the performance of automatic analysis. However, the accuracy of the manually corrected results is related to the quality of the automatic analysis result (Figure 3.24). When the automatic methods perform poorly, it would take a substantial amount of manual correction to reach a satisfactory accuracy. Since refining automatic methods is not the focus of eDetect, it would be ideal if eDetect could benefit from the advancements of the state-of-the-art automatic algorithms. At the moment it is possible to import segmentation and tracking results from CellProfiler [58,59] to eDetect. In the future we could also develop a FIJI [57] plugin based on eDetect.

Another limitation of eDetect is that it does not support segmentation of overlapping nuclei. For example, in Figure 3.20 C, D, F, and G, some nuclei overlap with each other - the pixels in the overlapped regions actually belong to two nuclei. But in eDetect, objects cannot overlap with each other - a pixel can only belong to at most one object. In these scenarios, there is no good way to divide the overlapping region with a single division line: both segmented objects will have to cover only a part of their own nuclei, and have to cover a part of each other's nuclei as well. This limitation largely hinders downstream analysis. It obviously makes morphological measurements inaccurate. It prevents automatic segmentation algorithm from noticing overlapping. It also makes automatic tracking more difficult. Nuclei overlapping is common with 2D widefield microscopy images. Supporting overlapping nuclei will greatly improve the usability of videos containing these events. Fortunately approaches tackling this problem already exist [146–148], and eDetect may integrate these approaches in the future.

Segmentation gating is aimed at separating different categories of segmented objects to the largest extent. However, the result of this PCA [141] based embedding always depends on the input data. With some datasets, there might be no clear boundaries between accurately segmented nuclei, inaccurately segmented nuclei, under-segmented objects, over-segmented objects, and other errors. In these cases the users would have to spend much time correctly selecting the target type of errors, or to fine-tune the 'formula' (Figure 3.13) to gain a better layout of the embedding. In the future, eDetect may integrate embedding methods like t-SNE [18] or UMAP [19] to provide users with more options, as long as the computational efficiency supports a real-time user experience.

For the current version, subsequent modules cell pair gating and cell lineages display are supposed to resolve the errors overlooked in segmentation gating. So the imperfection of segmentation gating itself does not affect the practicality of the

entire eDetect system. Quantitative evaluation also shows that in most datasets we have tried, eDetect has surpassed the best candidates of CTC [64, 65] regarding at least one of $F_1(CT)$ and $F_1(CL)$ within acceptable time (Tables 3.9, 3.10, and 3.11), and achieved a performance close to that of U-Net [4, 129] in dataset 'BBBC039' [130] (Figure 3.24) without using any training data. Based on these results we expect that eDetect will be very useful in many live cell imaging applications.

4.3 On human intervention

In live cell imaging data analysis, cell segmentation and cell tracking inevitably require human intervention to achieve high accuracy. Rule based methods approach high accuracy with iterative manual parameter tuning, which is based on human-eye inspection and evaluation of the results [58, 138]. Supervised learning based methods update parameters automatically, but need to be fed with annotated training data. Examples include ilastik [60], fastER [62], and U-Net [4, 66, 67]. However, as stated in the introduction, no approach is able to achieve perfect precision without post-edit, which is the third type of human intervention [1, 58, 59, 87, 89, 136, 137]. If a software tool provides enough degrees of freedom, and the user is willing to invest time, theoretically manual correction is able to achieve any level of accuracy and therefore leads to valid biological conclusions. As mentioned before, the main contribution of eDetect is to improve the efficiency of this inevitable post-edit step.

However, as we saw in the result part of this chapter, the manual correction time required for high accuracy depends on the quality of automatic analysis. Perhaps combining eDetect with supervised learning may further optimize the overall efficiency of the live cell imaging data analysis workflow. This could be a future direction of this work.

Bibliography

[1] Hongqing Han, Guoyu Wu, Yuchao Li, and Zhike Zi. eDetect: A Fast Error Detection and Correction Tool for Live Cell Imaging Data Analysis. *iScience*, 13:1–8, mar 2019.

[2] Robert Hooke. *Micrographia: or some physiological descriptions of minute bodies made by magnifying glasses, with observations and inquiries thereupon*. Courier Corporation, 2003.

[3] Patricia Fara. A microscopic reality tale. *Nature*, 459(7247):642–644, jun 2009.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. pages 234–241. 2015.

[5] Judith M S Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[6] Irwin Sobel. An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968*, 2014.

[7] Nobuyuki Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, jan 1979.

[8] Derek Bradley and Gerhard Roth. Adaptive Thresholding using the Integral Image. *Journal of Graphics Tools*, 12(2):13–21, jan 2007.

[9] Jean Serra and Pierre Soille. *Mathematical morphology and its applications to image processing*, volume 2. Springer Science & Business Media, 2012.

[10] Pierre Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[11] Béla Bollobás. *Modern graph theory*, volume 184. Springer Science & Business Media, 2013.

[12] Siegfried Weisenburger and Vahid Sandoghdar. Light microscopy: an ongoing contemporary revolution. *Contemporary Physics*, 56(2):123–143, apr 2015.

[13] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[14] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[15] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, jun 2014.

[16] Christian Wiwie, Jan Baumbach, and Richard Röttger. Comparing the performance of biomedical clustering methods. *Nature Methods*, 12(11):1033–1038, nov 2015.

[17] Hervé Abdi and Lynne J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, jul 2010.

[18] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579—-2605, 2008.

[19] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. feb 2018.

[20] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel W H Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature Biotechnology*, 37(1):38–44, jan 2019.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[23] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015.

[24] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular Systems Biology*, 12(7):878, jul 2016.

[25] Erick Moen, Dylan Bannon, Takamasa Kudo, William Graf, Markus Covert, and David Van Valen. Deep learning for cellular image analysis. *Nature Methods*, may 2019.

[26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, jun 2009.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, dec 2015.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[30] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. sep 2014.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper With Convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2015.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2016.

[33] Mingxing Tan and Quoc Le. {E}fficient{N}et: Rethinking Model Scaling

for Convolutional Neural Networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114, Long Beach, California, USA, 2019. PMLR.

[34] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. dec 2013.

[35] Thouis R Jones, In Kang, Douglas B Wheeler, Robert A Lindquist, Adam Papallo, David M Sabatini, Polina Golland, and Anne E Carpenter. CellProfiler Analyst: data exploration and analysis software for complex image-based screens. *BMC Bioinformatics*, 9(1):482, 2008.

[36] Pauli Rämö, Raphael Sacher, Berend Snijder, Boris Begemann, and Lucas Pelkmans. CellClassifier: supervised learning of cellular phenotypes. *Bioinformatics*, 25(22):3028–3030, nov 2009.

[37] Benjamin Misselwitz, Gerhard Strittmatter, Balamurugan Periaswamy, Markus C Schlumberger, Samuel Rout, Peter Horvath, Karol Kozak, and Wolf-Dietrich Hardt. Enhanced CellClassifier: a multi-class classification tool for microscopy images. *BMC Bioinformatics*, 11(1):30, dec 2010.

[38] Filippo Piccinini, Tamas Balassa, Abel Szkalisity, Csaba Molnar, Lassi Paavolainen, Kaisa Kujala, Krisztina Buzas, Marie Sarazova, Vilja Pietiainen, Ulrike Kutay, Kevin Smith, and Peter Horvath. Advanced Cell Classifier: User-Friendly Machine-Learning-Based Software for Discovering Phenotypes in High-Content Imaging Data. *Cell Systems*, 4(6):651–655.e5, jun 2017.

[39] Fei-Fei Li and P. Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE, 2005.

[40] Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2015.

[41] Oren Z. Kraus and Brendan J. Frey. Computer vision for high content screening. *Critical Reviews in Biochemistry and Molecular Biology*, 51(2):102–109, mar 2016.

[42] Oren Z. Kraus, Jimmy Lei Ba, and Brendan J. Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59, jun 2016.

[43] Oren Z Kraus, Ben T Grys, Jimmy Ba, Yolanda Chong, Brendan J Frey, Charles Boone, and Brenda J Andrews. Automated analysis of high-content microscopy data with deep learning. *Molecular Systems Biology*, 13(4):924, apr 2017.

[44] Devin P Sullivan, Casper F Winsnes, Lovisa Åkesson, Martin Hjelmare, Mikaela Wiking, Rutger Schutten, Linzi Campbell, Hjalti Leifsson, Scott Rhodes, Andie Nordgren, Kevin Smith, Bernard Revaz, Bergur Finnbogason, Attila Szantner, and Emma Lundberg. Deep learning is combined with massive-scale citizen science to improve large-scale image classification. *Nature Biotechnology*, 36(9):820–828, oct 2018.

[45] Gabriele Campanella, Matthew G. Hanna, Luke Geneslaw, Allen Miraflor, Vitor Werneck Krauss Silva, Klaus J. Busam, Edi Brogi, Victor E. Reuter, David S. Klimstra, and Thomas J. Fuchs. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. *Nature Medicine*, 25(8):1301–1309, aug 2019.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2015.

[47] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*, 111(1):98–136, jan 2015.

[48] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587. IEEE, jun 2014.

[49] Ross Girshick. Fast R-CNN. In *The IEEE International Conference on Computer Vision (ICCV)*, dec 2015.

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors,

Advances in Neural Information Processing Systems 28, pages 91–99. Curran Associates, Inc., 2015.

[51] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun 2016.

[52] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jul 2017.

[53] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. apr 2018.

[54] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *The IEEE International Conference on Computer Vision (ICCV)*, oct 2017.

[55] Erik Meijering. Cell Segmentation: 50 Years Down the Road [Life Sciences]. *IEEE Signal Processing Magazine*, 29(5):140–145, sep 2012.

[56] C. WAHLBY, I.-M. SINTORN, F. ERLANDSSON, G. BORGEFORS, and E. BENGTSSON. Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections. *Journal of Microscopy*, 215(1):67–76, jul 2004.

[57] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, 9(7):671–675, jul 2012.

[58] Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A Guertin, Joo Han Chang, Robert A Lindquist, Jason Moffat, Polina Golland, and David M Sabatini. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome biology*, 7(10):R100, 2006.

[59] Lee Kamentsky, Thouis R. Jones, Adam Fraser, Mark-Anthony Bray, David J. Logan, Katherine L. Madden, Vebjorn Ljosa, Curtis Rueden, Kevin W. Eliceiri, and Anne E. Carpenter. Improved structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. *Bioinformatics*, 27(8):1179–1180, apr 2011.

[60] Christoph Sommer, Christoph Straehle, Ullrich Kothe, and Fred A. Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 230–233. IEEE, mar 2011.

[61] David A. Van Valen, Takamasa Kudo, Keara M. Lane, Derek N. Macklin, Nicolas T. Quach, Mialy M. DeFelice, Inbal Maayan, Yu Tanouchi, Euan A. Ashley, and Markus W. Covert. Deep Learning Automates the Quantitative Analysis of Individual Cells in Live-Cell Imaging Experiments. *PLOS Computational Biology*, 12(11):e1005177, nov 2016.

[62] Oliver Hilsenbeck, Michael Schwarzfischer, Dirk Loeffler, Sotiris Dimopoulos, Simon Hastreiter, Carsten Marr, Fabian J Theis, and Timm Schroeder. fastER: a user-friendly tool for ultrafast and robust cell segmentation in large-scale microscopy. *Bioinformatics*, 33(13):2020–2028, jul 2017.

[63] Ignacio Arganda-Carreras, Srinivas C. Turaga, Daniel R. Berger, Dan Cireşan, Alessandro Giusti, Luca M. Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M. Buhmann, Ting Liu, Mojtaba Seyedhosseini, Tolga Tasdizen, Lee Kamentsky, Radim Burget, Vaclav Uher, Xiao Tan, Changming Sun, Tuan D. Pham, Erhan Bas, Mustafa G. Uzunbas, Albert Cardona, Johannes Schindelin, and H. Sebastian Seung. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9, nov 2015.

[64] Martin Maška, Vladimír Ulman, David Svoboda, Pavel Matula, Petr Matula, Cristina Ederra, Ainhoa Urbiola, Tomás España, Subramanian Venkatesan, Deepak M.W. Balak, Pavel Karas, Tereza Bolcková, Markéta Štreitová, Craig Carthel, Stefano Coraluppi, Nathalie Harder, Karl Rohr, Klas E. G. Magnusson, Joakim Jaldén, Helen M. Blau, Oleh Dzyubachyk, Pavel Křížek, Guy M. Hagen, David Pastor-Escuredo, Daniel Jimenez-Carretero, Maria J. Ledesma-Carbayo, Arrate Muñoz-Barrutia, Erik Meijering, Michal Kozubek, and Carlos Ortiz-de Solorzano. A benchmark for comparison of cell tracking algorithms. *Bioinformatics*, 30(11):1609–1617, jun 2014.

[65] Vladimír Ulman, Martin Maška, Klas E G Magnusson, Olaf Ronneberger, Carsten Haubold, Nathalie Harder, Pavel Matula, Petr Matula, David Svoboda, Miroslav Radojevic, Ihor Smal, Karl Rohr, Joakim Jaldén, Helen M Blau, Oleh Dzyubachyk, Boudewijn Lelieveldt, Pengdong Xiao, Yuexiang Li, Siu-Yeung Cho, Alexandre C Dufour, Jean-Christophe Olivo-Marin, Constantino C Reyes-Aldasoro, Jose A Solis-Lemus, Robert Bensch, Thomas

Brox, Johannes Stegmaier, Ralf Mikut, Steffen Wolf, Fred A Hamprecht, Tiago Esteves, Pedro Quelhas, Ömer Demirel, Lars Malmström, Florian Jug, Pavel Tomancak, Erik Meijering, Arrate Muñoz-Barrutia, Michal Kozubek, and Carlos Ortiz-de Solorzano. An objective comparison of cell-tracking algorithms. *Nature Methods*, 14(12):1141–1152, dec 2017.

[66] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. pages 424–432. 2016.

[67] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, Alexander Dovzhenko, Olaf Tietz, Cristina Dal Bosco, Sean Walsh, Deniz Saltukoglu, Tuan Leng Tay, Marco Prinz, Klaus Palme, Matias Simons, Ilka Diester, Thomas Brox, and Olaf Ronneberger. U-Net: deep learning for cell counting, detection, and morphometry. *Nature Methods*, 16(1):67–70, jan 2019.

[68] Florian Mueller, Adrien Senecal, Katjana Tantale, Hervé Marie-Nelly, Nathalie Ly, Olivier Collin, Eugenia Basyuk, Edouard Bertrand, Xavier Darzacq, and Christophe Zimmer. FISH-quant: automatic counting of transcripts in 3D FISH images. *Nature Methods*, 10(4):277–278, apr 2013.

[69] Nikolay Tsanov, Aubin Samacoits, Racha Chouaib, Abdel-Meneem Traboulsi, Thierry Gostan, Christian Weber, Christophe Zimmer, Kazem Zibara, Thomas Walter, Marion Peter, Edouard Bertrand, and Florian Mueller. smiFISH and FISH-quant – a flexible single RNA detection approach with super-resolution capability. *Nucleic Acids Research*, 44(22):e165–e165, dec 2016.

[70] Anne Trinh, Inga H Rye, Vanessa Almendro, Åslaug Helland, Hege G Russnes, and Florian Markowetz. GoIFISH: a system for the quantification of single cell heterogeneity from IFISH images. *Genome Biology*, 15(8):442, aug 2014.

[71] Allison Chia-Yi Wu and Scott A Rifkin. Aro: a machine learning approach to identifying single molecules and estimating classification error in fluorescence microscopy images. *BMC Bioinformatics*, 16(1):102, dec 2015.

[72] Emilio Maggio and Andrea Cavallaro. *Video tracking: theory and practice*. John Wiley & Sons, 2011.

[73] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854, dec 1979.

[74] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple Hypothesis Tracking Revisited. In *The IEEE International Conference on Computer Vision (ICCV)*, dec 2015.

[75] T. Fortmann, Y. Bar-Shalom, and M. Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE Journal of Oceanic Engineering*, 8(3):173–184, jul 1983.

[76] Seyed Hamid Rezatofighi, Anton Milan, Zhen Zhang, Qinfeng Shi, Anthony Dick, and Ian Reid. Joint Probabilistic Data Association Revisited. In *The IEEE International Conference on Computer Vision (ICCV)*, dec 2015.

[77] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, sep 2016.

[78] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[79] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, mar 1955.

[80] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, sep 2017.

[81] Erik Meijering, Oleh Dzyubachyk, Ihor Smal, and Wiggert A. van Cappellen. Tracking in cell and developmental biology. *Seminars in Cell & Developmental Biology*, 20(8):894–902, oct 2009.

[82] Erik Meijering, Oleh Dzyubachyk, and Ihor Smal. Methods for Cell and Particle Tracking. pages 183–200. 2012.

[83] Ricard Delgado-Gonzalo, Virginie Uhlmann, Daniel Schmitter, and Michael Unser. Snakes on a Plane: A perfect snap for bioimage analysis. *IEEE Signal Processing Magazine*, 32(1):41–48, jan 2015.

[84] Klas E. G. Magnusson and Joakim Jalden. A batch algorithm using iterative application of the Viterbi algorithm to track cells and construct cell lineages. In *2012 9th IEEE International Symposium on Biomedical Imaging (ISBI)*,

pages 382–385. IEEE, may 2012.

[85] Klas E. G. Magnusson, Joakim Jalden, Penney M. Gilbert, and Helen M. Blau. Global Linking of Cell Tracks Using the Viterbi Algorithm. *IEEE Transactions on Medical Imaging*, 34(4):911–929, apr 2015.

[86] Thomas A. Nketia, Heba Sailem, Gustavo Rohde, Raghu Machiraju, and Jens Rittscher. Analysis of live cell images: Methods, tools and opportunities. *Methods*, 115:65–79, feb 2017.

[87] Oliver Hilsenbeck, Michael Schwarzfischer, Stavroula Skylaki, Bernhard Schauberger, Philipp S Hoppe, Dirk Loeffler, Konstantinos D Kokkaliaris, Simon Hastreiter, Eleni Skylaki, Adam Filipczyk, Michael Strasser, Felix Buggenthin, Justin S Feigelman, Jan Krumsiek, Adrianus J J van den Berg, Max Endele, Martin Etzrodt, Carsten Marr, Fabian J Theis, and Timm Schroeder. Software tools for single-cell tracking and quantification of cellular and molecular properties. *Nature Biotechnology*, 34(7):703–706, jul 2016.

[88] Michael Held, Michael H A Schmitz, Bernd Fischer, Thomas Walter, Beate Neumann, Michael H Olma, Matthias Peter, Jan Ellenberg, and Daniel W Gerlich. CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging. *Nature Methods*, 7(9):747–754, sep 2010.

[89] Sam Cooper, Alexis R Barr, Robert Glen, and Chris Bakal. NucliTrack: an integrated nuclei tracking application. *Bioinformatics*, 33(20):3320–3322, oct 2017.

[90] Ricard Delgado-Gonzalo, Nicolas Denervaud, Sebastian Maerkl, and Michael Unser. Multi-target tracking of packed yeast cells. In *2010 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 544–547. IEEE, apr 2010.

[91] Khuloud Jaqaman, Dinah Loerke, Marcel Mettlen, Hirotaka Kuwata, Sergio Grinstein, Sandra L Schmid, and Gaudenz Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature Methods*, 5(8):695–702, aug 2008.

[92] Timm Schroeder. Long-term single-cell imaging of mammalian stem cells. *Nature Methods*, 8(S4):S30–S35, apr 2011.

[93] Martin Etzrodt, Max Endele, and Timm Schroeder. Quantitative Single-Cell Approaches to Stem Cell Research. *Cell Stem Cell*, 15(5):546–558, nov 2014.

[94] Philipp S. Hoppe, Daniel L. Coutu, and Timm Schroeder. Single-cell technologies sharpen up mammalian stem cell research. *Nature Cell Biology*, 16(10):919–927, oct 2014.

[95] Ralph A. Bradshaw and Edward A. Dennis, editors. *Handbook of Cell Signaling*. Academic Press, 2003.

[96] Xin-Hua Feng and Rik Derynck. SPECIFICITY AND VERSATILITY IN TGF-β SIGNALING THROUGH SMADS. *Annual Review of Cell and Developmental Biology*, 21(1):659–693, nov 2005.

[97] Caroline S Hill. Nucleocytoplasmic shuttling of Smad proteins. *Cell Research*, 19(1):36–46, jan 2009.

[98] Joan Massagué. TGFβ signalling in context. *Nature Reviews Molecular Cell Biology*, 13(10):616–630, oct 2012.

[99] Agata Zieba, Katerina Pardali, Ola Söderberg, Lena Lindbom, Erik Nyström, Aristidis Moustakas, Carl-Henrik Heldin, and Ulf Landegren. Intercellular Variation in Signaling through the TGF-β Pathway and Its Relation to Cell Density and Cell Cycle Phase. *Molecular & Cellular Proteomics*, 11(7):M111.013482, jul 2012.

[100] A. Warmflash, Q. Zhang, B. Sorre, A. Vonica, E. D. Siggia, and A. H. Brivanlou. Dynamics of TGF- signaling reveal adaptive and pulsatile behaviors reflected in the nuclear localization of transcription factor Smad4. *Proceedings of the National Academy of Sciences*, 109(28):E1947–E1956, jul 2012.

[101] Jette Strasen, Uddipan Sarma, Marcel Jentsch, Stefan Bohn, Caibin Sheng, Daniel Horbelt, Petra Knaus, Stefan Legewie, and Alexander Loewer. Cell-specific responses to the cytokine TGFβ are determined by variability in protein levels. *Molecular Systems Biology*, 14(1):e7733, jan 2018.

[102] Christopher L. Frick, Clare Yarka, Harry Nunns, and Lea Goentoro. Sensing relative signal in the Tgf-β/Smad pathway. *Proceedings of the National Academy of Sciences*, 114(14):E2975–E2982, apr 2017.

[103] David Owen Morgan. *The Cell Cycle: Principles of Control*. New Science Press Ltd, 2007.

[104] Asako Sakaue-Sawano, Hiroshi Kurokawa, Toshifumi Morimura, Aki Hanyu,

Hiroshi Hama, Hatsuki Osawa, Saori Kashiwagi, Kiyoko Fukami, Takaki Miyata, Hiroyuki Miyoshi, Takeshi Imamura, Masaharu Ogawa, Hisao Masai, and Atsushi Miyawaki. Visualizing Spatiotemporal Dynamics of Multicellular Cell-Cycle Progression. *Cell*, 132(3):487–498, feb 2008.

[105] S. Lim and P. Kaldis. Cdks, cyclins and CKIs: roles beyond cell cycle regulation. *Development*, 140(15):3079–3093, aug 2013.

[106] Todd Waldman, Kenneth W Kinzler, and Bert Vogelstein. p21 Is Necessary for the p53-mediated G1 Arrest in Human Cancer Cells. *Cancer Research*, 55(22):5187–5190, 1995.

[107] James Brugarolas, Chitra Chandrasekaran, Jeffrey I. Gordon, David Beach, Tyler Jacks, and Gregory J. Hannon. Radiation-induced cell cycle arrest compromised by p21 deficiency. *Nature*, 377(6549):552–557, oct 1995.

[108] F. Bunz. Requirement for p53 and p21 to Sustain G2 Arrest After DNA Damage. *Science*, 282(5393):1497–1501, nov 1998.

[109] Alexander Loewer, Eric Batchelor, Giorgio Gaglia, and Galit Lahav. Basal Dynamics of p53 Reveal Transcriptionally Attenuated Pulses in Cycling Cells. *Cell*, 142(1):89–100, jul 2010.

[110] Sabrina L. Spencer, Steven D. Cappell, Feng-Chiao Tsai, K. Wesley Overton, Clifford L. Wang, and Tobias Meyer. The Proliferation-Quiescence Decision Is Controlled by a Bifurcation in CDK2 Activity at Mitotic Exit. *Cell*, 155(2):369–383, oct 2013.

[111] Mansi Arora, Justin Moser, Harsha Phadke, Ashik Akbar Basha, and Sabrina L. Spencer. Endogenous Replication Stress in Mother Cells Leads to Quiescence of Daughter Cells. *Cell Reports*, 19(7):1351–1364, may 2017.

[112] Alexis R. Barr, Samuel Cooper, Frank S. Heldt, Francesca Butera, Henriette Stoy, Jörg Mansfeld, Béla Novák, and Chris Bakal. DNA damage during S-phase mediates the proliferation-quiescence decision in the subsequent G1 via p21 expression. *Nature Communications*, 8(1):14728, apr 2017.

[113] José Reyes, Jia-Yun Chen, Jacob Stewart-Ornstein, Kyle W. Karhohs, Caroline S. Mock, and Galit Lahav. Fluctuations in p53 Signaling Allow Escape from Cell-Cycle Arrest. *Molecular Cell*, 71(4):581–591.e5, aug 2018.

[114] Hui Xiao Chao, Cere E. Poovey, Ashley A. Privette, Gavin D. Grant, Hui Yan

Chao, Jeanette G. Cook, and Jeremy E. Purvis. Orchestration of DNA Damage Checkpoint Dynamics across the Human Cell Cycle. *Cell Systems*, 5(5):445–459.e5, nov 2017.

[115] Edouard Bertrand, Pascal Chartrand, Matthias Schaefer, Shailesh M. Shenoy, Robert H. Singer, and Roy M. Long. Localization of ASH1 mRNA Particles in Living Yeast. *Molecular Cell*, 2(4):437–445, oct 1998.

[116] D. R. Larson, D. Zenklusen, B. Wu, J. A. Chao, and R. H. Singer. Real-Time Observation of Transcription Initiation and Elongation on an Endogenous Yeast Gene. *Science*, 332(6028):475–478, apr 2011.

[117] Ido Golding, Johan Paulsson, Scott M. Zawilski, and Edward C. Cox. Real-Time Kinetics of Gene Activity in Individual Bacteria. *Cell*, 123(6):1025–1036, dec 2005.

[118] Jonathan R. Chubb, Tatjana Trcek, Shailesh M. Shenoy, and Robert H. Singer. Transcriptional Pulsing of a Developmental Gene. *Current Biology*, 16(10):1018–1025, may 2006.

[119] Katjana Tantale, Florian Mueller, Alja Kozulic-Pirher, Annick Lesne, Jean-Marc Victor, Marie-Cécile Robert, Serena Capozi, Racha Chouaib, Volker Bäcker, Julio Mateos-Langerak, Xavier Darzacq, Christophe Zimmer, Eugenia Basyuk, and Edouard Bertrand. A single-molecule view of transcription reveals convoys of RNA polymerases and multi-scale bursting. *Nature Communications*, 7(1):12248, nov 2016.

[120] Takashi Fukaya, Bomyi Lim, and Michael Levine. Enhancer Control of Transcriptional Bursting. *Cell*, 166(2):358–368, jul 2016.

[121] W Su, S Jackson, R Tjian, and H Echols. DNA looping between sites for transcriptional activation: self-association of DNA-bound Sp1. *Genes & Development*, 5(5):820–826, may 1991.

[122] Yad Ghavi-Helm, Felix A. Klein, Tibor Pakozdi, Lucia Ciglar, Daan Noordermeer, Wolfgang Huber, and Eileen E. M. Furlong. Enhancer loops appear stable during development and are associated with paused polymerase. *Nature*, 512(7512):96–100, aug 2014.

[123] Hongtao Chen, Michal Levo, Lev Barinov, Miki Fujioka, James B. Jaynes, and Thomas Gregor. Dynamic interplay between enhancer–promoter topology and gene activity. *Nature Genetics*, 50(9):1296–1303, sep 2018.

[124] Clemens B. Hug, Alexis G. Grimaldi, Kai Kruse, and Juan M. Vaquerizas. Chromatin Architecture Emerges during Zygotic Genome Activation Independent of Transcription. *Cell*, 169(2):216–228.e19, apr 2017.

[125] Adam J Rubin, Brook C Barajas, Mayra Furlan-Magaril, Vanessa Lopez-Pajares, Maxwell R Mumbach, Imani Howard, Daniel S Kim, Lisa D Boxer, Jonathan Cairns, Mikhail Spivakov, Steven W Wingett, Minyi Shi, Zhixin Zhao, William J Greenleaf, Anshul Kundaje, Michael Snyder, Howard Y Chang, Peter Fraser, and Paul A Khavari. Lineage-specific dynamic and pre-established enhancer–promoter contacts cooperate in terminal differentiation. *Nature Genetics*, 49(10):1522–1528, oct 2017.

[126] Eva Bártová, Gabriela Šustáčková, Lenka Stixová, Stanislav Kozubek, Soňa Legartová, and Veronika Foltánková. Recruitment of Oct4 Protein to UV-Damaged Chromatin in Embryonic Stem Cells. *PLoS ONE*, 6(12):e27281, dec 2011.

[127] Beate Neumann, Thomas Walter, Jean-Karim Hériché, Jutta Bulkescher, Holger Erfle, Christian Conrad, Phill Rogers, Ina Poser, Michael Held, Urban Liebel, Cihan Cetin, Frank Sieckmann, Gregoire Pau, Rolf Kabbe, Annelie Wünsche, Venkata Satagopam, Michael H. A. Schmitz, Catherine Chapuis, Daniel W. Gerlich, Reinhard Schneider, Roland Eils, Wolfgang Huber, Jan-Michael Peters, Anthony A. Hyman, Richard Durbin, Rainer Pepperkok, and Jan Ellenberg. Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. *Nature*, 464(7289):721–727, apr 2010.

[128] David Svoboda and Vladimir Ulman. MitoGen: A Framework for Generating 3D Synthetic Time-Lapse Sequences of Cell Populations in Fluorescence Microscopy. *IEEE Transactions on Medical Imaging*, 36(1):310–321, jan 2017.

[129] Juan C Caicedo, Jonathan Roth, Allen Goodman, Tim Becker, Kyle W Karhohs, Matthieu Broisin, Molnar Csaba, Claire McQuin, Shantanu Singh, Fabian Theis, and Others. Evaluation of deep learning strategies for nucleus segmentation in fluorescence images. *BioRxiv*, page 335216, 2019.

[130] Vebjorn Ljosa, Katherine L Sokolnicki, and Anne E Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature Methods*, 9(7):637–637, jul 2012.

[131] Pavel Matula, Martin Maška, Dmitry V. Sorokin, Petr Matula, Carlos Ortiz-de Solórzano, and Michal Kozubek. Cell Tracking Accuracy Measurement Based on Comparison of Acyclic Oriented Graphs. *PLOS ONE*,

10(12):e0144959, dec 2015.

[132] Kang Li, Eric D. Miller, Mei Chen, Takeo Kanade, Lee E. Weiss, and Phil G. Campbell. Cell population tracking and lineage construction with spatiotemporal context. *Medical Image Analysis*, 12(5):546–566, oct 2008.

[133] Paul Jaccard. THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. *New Phytologist*, 11(2):37–50, feb 1912.

[134] Yutaka Sasaki. The truth of the F-measure. 2007.

[135] Stavroula Skylaki, Oliver Hilsenbeck, and Timm Schroeder. Challenges in long-term imaging and quantification of single-cell dynamics. *Nature Biotechnology*, 34(11):1137–1144, nov 2016.

[136] Filippo Piccinini, Alexa Kiss, and Peter Horvath. CellTracker (not only) for dummies. *Bioinformatics*, 32(6):955–957, mar 2016.

[137] Mark Winter, Walter Mankowski, Eric Wait, Sally Temple, and Andrew R. Cohen. LEVER: software tools for segmentation, tracking and lineaging of proliferating cells. *Bioinformatics*, page btw406, jul 2016.

[138] Mark-Anthony Bray and Anne E. Carpenter. CellProfiler Tracer: exploring and validating high-throughput, time-lapse microscopy image data. *BMC Bioinformatics*, 16(1):369, dec 2015.

[139] Robert M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(6):610–621, nov 1973.

[140] Michael V. Boland, Mia K. Markey, and Robert F. Murphy. Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images. *Cytometry*, 33(3):366–375, nov 1998.

[141] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[142] Fernand Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113–125, jul 1994.

[143] Nima Moshtagh. Minimum volume enclosing ellipsoid. *Convex Optimization*, 111(January):1—-9, 2005.

[144] Gang Lin, Monica K. Chawla, Kathy Olson, John F. Guzowski, Carol A. Barnes, and Badrinath Roysam. Hierarchical, model-based merging of multiple fragments for improved three-dimensional segmentation of nuclei. *Cytometry Part A*, 63A(1):20–33, jan 2005.

[145] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, jul 2013.

[146] Csaba Molnar, Ian H. Jermyn, Zoltan Kato, Vesa Rahkama, Päivi Östling, Piia Mikkonen, Vilja Pietiäinen, and Peter Horvath. Accurate Morphology Preserving Segmentation of Overlapping Cells based on Active Contours. *Scientific Reports*, 6(1):32412, sep 2016.

[147] Anton Bohm, Annekathrin Ucker, Tim Jager, Olaf Ronneberger, and Thorsten Falk. ISOO DL: Instance segmentation of overlapping biological objects using deep learning. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 1225–1229. IEEE, apr 2018.

[148] Anton Bohm, Maxim Tatarchenko, and Thorsten Falk. ISOO V2 DL - Semantic Instance Segmentation of Touching and Overlapping Objects. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 343–347. IEEE, apr 2019.

List of Tables

List of Figures

Glossary

CMOS complementary metal–oxide–semiconductor. 21

CCD charge-coupled device. 21

CNN convolutional neural network. 25, 26, 28

SVM support vector machine. 28

FISH Fluorescence *in situ* hybridization. 29

CTC Cell Tracking Challenge. 30, 39, 49, 55, 59, 81, 92, 94

RNA ribonucleic acid. 32, 35

DNA deoxyribonucleic acid. 32–35

TGF-β transforming growth factor β. 32, 33

G1 gap 1 (phase). 33–35

S synthesis (phase). 33–35

G2 gap 2 (phase). 33–35

M mitosis (phase). 33, 34

G0 gap 0 (phase). 33

FUCCI fluorescent ubiquitination-based cell-cycle indicator. 34, 35, 39

GFP green fluorescent protein. 34, 35

RFP red fluorescent protein. 34

CDK cyclin-dependent kinase. 34, 35

CKI cyclin-dependent kinases inhibitor. 34

mRNA messenger RNA. 35

MCP MS2 coat protein. 35

PCP PP7 coat protein. 35

HIV human immunodeficiency virus. 35

2D 2-dimensional. 38, 61, 64–66, 93

CFP cyan fluorescent protein. 39

H2B histone H2B. 39

Pr prediction. 49

GT ground truth. 49, 51

SEG segmentation accuracy. 49, 51, 54, 55, 58, 60, 81, 84, 92

TRA tracking accuracy. 49, 53, 55, 60, 81, 84, 92

CT complete track. 49, 53, 54, 71, 92

CL complete lineage. 49, 53, 54, 59, 71, 74, 92

IoU intersection over union. 49, 54, 58, 85

TP true positive. 53, 54

TN true negative. 53

FP false positive. 53, 54

FN false negative. 53, 54

GUI graphical user interface. 62, 64, 79, 92

PCA principal component analysis. 66, 70, 93

PC principal component. 66, 70

Acknowledgements

First and foremost I would like to express my deep gratitude to my supervisor Dr. Zhike Zi for guiding me into a field that matches my background and meets my curiosity, and for showing me this wonderful microscopic world which I loved more and more. I am also grateful for his patient guidance and enthusiastic encouragements in countless discussions throughout the years. I would like to express my very great appreciation to both Prof. Edda Klipp and Dr. Edda Schulz for giving helpful feedbacks and suggestions at my TAC meetings. In addition, I want to thank Prof. Klipp for kindly being my university supervisor and introducing to me the science in her group. I also want to thank my officemates Dr. Marijn van Jaarsveld, Dr. Yuchao Li, Dr. Guoyu Wu and Dan Shi, with whom I have had the pleasure to work. Marijn brought me to see the very inspiring FACS system. Yuchao and Guoyu let me come along while they operated on the microscopes. These efforts are much appreciated. I also want to thank all of them for the discussions from which I learned a lot of biology. I wish to acknowledge the financial support from IMPRS-CBSC, without which the work described in this thesis would not have been possible. I also want to thank Dr. Kirsten Kelleher and Dr. Fabian Feutlinske, for all the help with issues regarding IMPRS and the doctorate. Special thanks should be given to Dr. Zi, Kirsten and Jörg Winkler for generously sparing time to proofread this thesis.

In the past few months I have been trying to figure out my future. Apart from the support and advices from Dr. Zi, many other people helped me gain perspectives. Dr. Rene Buschow kindly showed me what he had been doing and reading. I heard of FISH for the first time from Verena Mutzel, who shared a lot of microscopy images and her view on image analysis. Since then I started reading papers on related topics. Martyna Gajos also shared many interesting papers and ideas. Prof. Ana Pombo warmheartedly provided the opportunity of collaboration. It is my pleasure to work and discuss with Silvia Carvalho from Pombo lab. I also want to thank the members of Schulz lab, from whom I learned a lot of biology while participating in their journal club. I want to say thank you to all these people,

whose enthusiastic help meant a lot to me.

I would like to offer my special thanks to the people in the IT department, especially Donald Buczek, for the consistent technical support throughout the PhD.

In the end, I want to thank my family, especially my parents, for their unconditional love.

Declaration

I hereby declare that I completed the doctoral thesis independently based on the stated resources and aids. I have not applied for a doctoral degree elsewhere and do not have a corresponding doctoral degree. I have not submitted the doctoral thesis, or parts of it, to another academic institution and the thesis has not been accepted or rejected. I declare that I have acknowledged the Doctoral Degree Regulations which underlie the procedure of the Faculty of Life Sciences of Humboldt-Universität zu Berlin, as amended on 5th March 2015. Furthermore, I declare that no collaboration with commercial doctoral degree supervisors took place, and that the principles of Humboldt-Universität zu Berlin for ensuring good academic practice were abided by.

Date _____

Signature _____